Determination Method of Evolution Type of Service-oriented Entity

Dianlong You, Limin Shen
School of Information Science and Engineering, Yanshan University, Qinhuangdao, 066004, P. R. China
Email: dianlongyou@gmail.com, shenllmm@sina.com

Abstract—The determination of evolution type plays an important role in securing the accurate and efficient evolution of service composition. Based on the applied level in the evolution of service composition, using open-bisimulation theory, evolution type can be decided from three dimensions, which are respectively single state node, single service and service composition, and the determination rule as well. The determination of evolution type for single state node can tell whether the external behaviors of the node has changed, while the single service can confirm the changes that the state nodes of the service have had in quantity and logic during the business process, and the evolution type of service composition can provide guidance for fixing the final evolution program. Finally, a case study of how to decide the evolution type of web service composition is off ered to demonstrate the practical application of the above rules.

Index Terms—service composition, service choreography, service evolution, evolution type, determination method

I. INTRODUCTION

The Web service composition indicates to enable multiple independent and autonomous Web services to communicate and coordinate by the syntax and logical relation for specific target and realize specific service function. To adapt the environment change and continuously meet the user requirement, the Web service composition should be adjusted and can evolve [1]. The web service composition evolves in the service function and service performance, including service architecture, service logic, session protocol, security mechanism and communication rule change. The service process evolution focuses on the service logic and rule of the service composition, belongs to function evolution and is the elementary problem in the service composition evolution. Now the participation service in Web service composition evolves in the service function, evolution type, determination method.

service choreography describes the interaction protocol between the services[3-4]. The evolution at the service orchestration level occurs inside single service, which is invisible to exterior and will not affect the whole such as correction and optimization of single service. The evolution at the service Choreography breaks through single service and penetrates into the copartner service. The evolution process requires coordination and participation of the copartner service with adjustment of the service choreography protocol [3].

II. DEMANDS AND ISSUES

Determination of the evolution type is meaningful for improving correctness and efficiency of the big-granularity service composition evolution. The logic relation among the status nodes can be identified via analysis on evolution type to reduce the logic confliction and business flow error in the service composition evolution design phase. Before the service composition evolution is implemented, determining the evolution type can reduce the evolution cost and improve evolution efficiency. E.g. the value-added evolution can change the business flow via extension of the old service choreography protocol. The reduced evolution can directly delete the business flow inside the services without change of the old service choreography protocol. The evolution type is determined after the evolution is implemented (especially dynamic evolution) to get the change before and after service composition evolution and record the evolution process via the evolution log.

Now the service composition evolution is studied at the service orchestration level. The collaborative evolution of multiple services at the service choreography level is the difficulty in the service composition evolution all the time. [3] This study focuses on determination of the evolution type of big-granularity Web service composition business flow at the service choreography service in case of evolution and regards the evolution at the service Orchestration as a special case of the evolution at the service Choreography level. This study mainly determines the evolution type based on the bisimulation theory, shown as the Figure 1. (a) Determine evolution type of single state node. Based on the weak simulation and weak bisimulation theory, the evolution of the state nodes are divided into the observational equivalence, added value, reduction and change and the
determination rule is given; (b) Determine evolution type of single service: based on strong simulation and weak simulation theory, the evolution type of single service is divided into the elementary action type and compound action type by the change of state node prior to and after the business flow evolution inside the service. The elementary action includes addition, deletion and change operation of the state nodes. The compound action type includes continued consolidation, parallel consolidation, continued decomposition and parallel decomposition. The corresponding determination rule is given. (c) Determine evolution type of service composition: the evolution type of web service composition is divided into internal type, value-added type, reduced type and global evolution type by the change of the global business flow prior to and after evolution and the determination rules are given.

![Figure 1. Determination of evolution type](image1)

### III. RELATED WORK

The reference [4] divides the Web service composition into service orchestration and choreography. The service orchestration describes the business flow execution in the service composition. The service choreography describes the interaction protocol between services. The reference [3] considers the case that the local service orchestration evolution can not affect other copartner services in the service choreography, namely it guarantees that the evolution is transparent to the copartner service and the evolution belongs to the service orchestration level. Its influence is only restricted in single service and other copartner services are not evolved. The reference [1] proposes that the service composition evolution can occur at the process definition level and process instance level. The evolution at the process definition level will change WS-CDL or BPEL procedure and the evolution affects all Web service composition instances. The evolution at the process instance level only changes the process definition of the special service composition instance and only changed service composition instance is affected. The reference [6] gives the dynamic evolution model of the service choreography and gives the method for the service choreography transmission action, confirmation rule permitted by service Choreography and mutex solution. The service influence of this model only involves determination of the influence scope of originated service. Other service change caused by the copartner service evolution is not analyzed.

The reference [7] proposes the dynamic evolution method supporting Web service business flow protocol at the business flow level. The service protocol evolution is divided into full replacement, state replacement and path replacement, history-based replacement and customer protocol analysis replacement and the rules and algorithms are given. The reference [8] gives shallow evolution and deep evolution. The shallow evolution only involves single service. The deep evolution breaks throughput single service and penetrates into the copartner service, so it generate the cascading evolution. The reference [9] thinks that the choreography implementability and service orchestration compliance to a service Choreography is the key in the service Choreography evolution.

In a word, the features of related research work in the service composition evolution at home and abroad are described as follows: (a) The evolution mainly concentrates at the service orchestration level or shallow evolution, namely the evolution mainly occurs inside single service and does not involve other participation services; (b) The evolution belongs to the process instance level and only affects current Web service composition instance; (c) The evolution method at the service Choreography level mainly performs full replacement, state replacement and path replacement of the business flow by the service Choreography protocol. The research on the determination method of the evolution type under the big-granularity service composition is limited.

### IV. CONCEPTS AND MODELS

**Definition 1 (state transition model).** One state transition system in action set Active is a two-tuple \((state, relation)\), wherein:

- \(active\) is an action set. The action is defined as one execution which causes the service state change.
- \(state\) is a state set.
- \(relation\) indicates a transition relation, \(relation \subseteq state \times active \times state\). If \(a \in active\), \(s, s' \in state\) and \((s,a,s') \in relation\) is met, it is marked as \(s \xrightarrow{a} s'\).

**Definition 2 (business flow).** The business flow \(b\) is the state transition from the start state \(s_i\) to the ending state \(s_j\) caused by the action, namely \(s_i \xrightarrow{a_1} s_{i1} \xrightarrow{a_2} \cdots \xrightarrow{a_n} s_j\), it is marked as \((s_i, s_{i1}, \cdots, s_j)\) or \((s_i \triangleright s_j)\). \(fn(b)\) indicates the set of the state in \(b\). Generally \(s_0\) indicates initial state. \(s_i\) indicates the ending state. For the state sequence \((s_i, s_{i1}, \cdots, s_j)\), \(s_0 = s_i\) and \(s_j = s_j\). Multiple business flows of the service composition may be available. They are named as \(b_1, b_2, \ldots, b_n\) in turn. The state node of single business flow \(b_i\) of the service composition may distribute inside the service. If \(service\) includes the state node of \(b_i\), the state transition sequences composed by these nodes (single node is possible) are called as \(b_i\) and exists in the internal business flow of \(service\). Multiple such internal business flows may exist in \(service\).
internal business flow is marked as \( b_{00} \), \( b_{01} \) indicates \( k \) th internal business flow of the business flow \( b_j \) in the service \( \text{service} \).

Definition 3 (WSC of Web service composition). it is a two tuple \( \text{WSC} \triangleq \text{Service, Bp, p} > \), wherein:
- \( \text{Service} \triangleq \{ \text{service}, \text{service}, \ldots, \text{service} \} \) is the set of all copartner services in the service composition and is called as the service set. \( n \) indicates the number of the copartner services in the service composition.
- \( \text{Bp} \triangleq \{ b_1, b_2, \ldots, b_n \} \) is the set of all business flows of the service composition. \( m \) indicates the number of the business flow.
- \( p \) is the service choreography protocol.

Definition 4 (strong simulation). assume that \((\text{state, relation})\) is a state transition system. If \( s_p \cdot s_p \cdot s_q \cdot s_q \in \text{state} \) and \( \mathcal{R} = \{(s_p \cdot s_p'),(s_q \cdot s_q'),\ldots\} \) is two tuple relation in \text{state}. For any \( s_p \cdot \mathcal{R} \cdot s_q \), the following conditions are met:

If \( s_p \xrightarrow{\alpha} s_p' \), \( s_q \) is available to make \( s_q \xrightarrow{\alpha} s_q' \) and \( s_p \cdot \mathcal{R} \cdot s_q \), \( \mathcal{R} \) is called as a strong simulation in \((\text{state, relation})\). If a strong simulation \( \mathcal{R} \) makes \( s_p \cdot \mathcal{R} \cdot s_q \), \( s_q \) is called as strong simulation \( s_p \).

Definition 5 (strong bismulation and strong equivalence). if two duple relation \( \mathcal{R} \) and its reverse \( \mathcal{R}^{-1} \) is the strong simulation of the state transition system \((\text{state, relation})\), \( s_p, s_q \in \text{state} \) and \( \mathcal{R} = \{(s_p \cdot s_p'),(s_q \cdot s_q'),\ldots\} \) is two duple relation in \text{state}. If a strong simulation \( \mathcal{R} \) makes \( s_p \cdot \mathcal{R} \cdot s_q \), two state \( s_p \) and \( s_q \) are strong bismulation or strong equivalence. It is marked as \( s_p \sim s_q \).

Definition 6 (weak simulation). assume that \((\text{state, relation})\) is a state transition system, \( s_p, s_q, s_p, s_q \in \text{state} \) and \( \mathcal{R} = \{(s_p \cdot s_p'),(s_q \cdot s_q'),\ldots\} \) is two tuple relation in \text{state}. For any \( s_p \cdot \mathcal{R} \cdot s_q \), the following condition is met:

If \( s_p \xrightarrow{\epsilon} s_p' \), \( s_q \) makes \( s_q \xrightarrow{\epsilon} s_q' \) and \( s_p \cdot \mathcal{R} \cdot s_q \), \( S \) is called as one weak simulation in \((\text{state, relation})\). If one weak simulation \( S \) makes \( s_p \cdot \mathcal{R} \cdot s_q \), \( s_q \) is the weak simulation of \( s_p \). \( \epsilon \) indicates visible action sequence \( \lambda_1 \cdots \lambda_n \). \( s_p \xrightarrow{\epsilon} s_q \) indicates zero or multiple internal actions.

Definition 7 (weak bismulation, weak equivalence or observational equivalence). if two tuple relation \( S \) and its reverse \( S^{-1} \) is the weak simulation of the state transition system \((\text{state, relation})\). \( s_p, s_q \in \text{state} \), \( S = \{(s_p \cdot s_p'),(s_q \cdot s_q'),\ldots\} \) is a two-tuple relation in \text{state}. If one weak simulation \( S \) makes \( s_p \cdot \mathcal{R} \cdot s_q \), the state \( s_p \) and \( s_q \) are called as weak bismulation, weak equivalence or observational equivalence and it is marked as \( s_p \approx s_q \).

The bismulation equivalence is the common equivalence relation in the process algebra and is divided into the strong bismulation equivalence and weak bismulation equivalence by internal action ignorance or not. The strong bismulation regards all actions (including invisible actions inside) equally. The weak bismulation ignores the internal actions and can observe equivalence of the internal actions.

V. Determination of Service Composition Evolution Type

The evolution type determination is the foundation for identifying the influence range of the service composition business flow evolution. The service composition evolution may only include addition, deletion or change of business flows inside single service between services, or re-orchestration of the business flows inside the service and re-choreography of the interaction protocol among the services \([6, 10]\), so identifying the evolution rule by the evolution type can improve the evolution efficiency and reduce evolution cost.

A. Determination of State Node Evolution Type

The state node indicates the function points inside the services which state change is caused by the input action, E.g. component and function module. evolution of single state node includes increase, decrease and adjustment. “Value-added” indicates the function extension of the state node without change of old functions. “Reduced” indicates decrease of old functions and no new functions. “Adjusted” indicates deletion of old functions and addition of new functions. The rule 1 utilizes the weak simulation and weak bismulation in the bismulation theory, does not consider the behaviors inside the state node and only determines the observational behaviors outside the state nodes.

Rule 1 (determination of state node evolution type). \( p \) is a state node of service. After \( p \) evolution, the state node is called as \( p' \). \( s_p \) is the state set of \( p \). \( s_{p'} \) is the state set of \( p' \). Prior to and after \( p \) evolution, the state transition model is \( (s_p, r_p) \) and \( (s_{p'}, r_{p'}) \) respectively. If \( s \in s_p, s' \in s_{p'} \), the determination rule is described as follows.

- For \( \forall s, \exists s' \) and \( \forall s', \exists s \), if \( s Ss' \wedge s' Ss \) (it is also marked as \( s = s' \)), namely \( s \) is weak equivalent to \( s' \). It indicates that \( p \) does not evolve from external behavior (\( p \) may evolve inside) and belongs to observational equivalence.
- If \( s Ss' \wedge s' Ss \), namely \( s' \) is weak simulation of \( s \) and \( s \) is not weak simulation of \( s' \), it indicates that \( p' \) includes all observational
behaviors of \( p \) and \( p' \) does not include the state transition, namely the observational behaviors adds value when \( p \) evolves to \( p' \).

- If \( sSs \wedge s'Fs' \), namely \( s \) is weak simulation of \( s' \) and \( s' \) is not weak simulation of \( s \), it indicates that \( p' \) includes partial observational behaviors of \( p \) and \( p \) does not include all observational behaviors of \( p' \), namely the observational behaviors reduce when \( p \) evolves to \( p' \).

- If \( sFs' \wedge sFs \) (it is also called as \( s \not\equiv s' \)), namely \( s \) is not weak equivalence of \( s' \). It indicates that \( p \) and \( p' \) include the different external behaviors, namely the observational behaviors change when \( p \) evolves to \( p' \).

The rule 1 does not focus on the behaviors inside the state nodes. The external behaviors of the state nodes can be determined by using weak bisimulation method prior to and after evolution, namely change of the node state transition sequence, so the evolution type of single state node can be identified. E.g. The state node \( p \) and the evolved state node \( p' \) are shown as the Figure 2 and Figure 3. If the state transition sequence is \( \cdots s_j \xrightarrow{a_{ij}} s_{i1} \xrightarrow{a_{i1}} \cdots \xrightarrow{a_{ik}} s_{i} \cdots \) prior to evolution, the TABLE 1 gives the mapping of the state transition sequence and evolution type after evolution.

![Figure 2. State node prior to evolution](image1)

![Figure 3. State node after evolution](image2)

TABLE 1. RELATION BETWEEN STATE TRANSITION SEQUENCE AND EVOLUTION TYPE OF THE STATE NODE \( p \)

<table>
<thead>
<tr>
<th>state transition sequence after the state node ( p ) evolves to ( p' )</th>
<th>Evolution type</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \cdots s_j \xrightarrow{a_{ij}} s_{i1} \xrightarrow{a_{i1}} \cdots \xrightarrow{a_{ik}} s_{i} \cdots )</td>
<td>observational equivalence</td>
</tr>
<tr>
<td></td>
<td>added</td>
</tr>
<tr>
<td></td>
<td>reduced</td>
</tr>
<tr>
<td></td>
<td>changed</td>
</tr>
</tbody>
</table>

B. Determination of Evolution Type of Single Service

Single service evolution focuses on the change of the external service behaviors caused by change of state node quantity and the logic relation in the business flow in the services. Evolution type of single service includes the basic action type and combined action type. The basic action includes addition, deletion and change of the state nodes. “Add” indicates to add new state nodes. “Delete” indicates to delete old state nodes. “Change” indicates to add, reduce or adjust the old state nodes. The determination method is shown as the rule 2.

Rule 2 (determination of evolution of elementary action type in single service). \( p \) is the state node in service \( s \). \( q \) is the state node in the service' after evolution. \( s' \) is the state set of \( q \). state and state' are the state set of service and service'. \( s'p \subseteq state \) and \( s' \subseteq state' \). The determination rule is described as follows: for \( \forall s' \in s_q, \forall s \in state \),

- If \( \exists q \) and \( sFs' \wedge sFs \), namely \( s \) is not weak equivalence to \( s' \), it indicates that service does not include the state node to simulate \( p' \), namely new state nodes are added in evolution from service to service'.

- If \( \forall q \) and \( sRs \wedge sFs' \), namely \( s \) is strong simulation of \( s' \) and \( s' \) is not weak simulation of \( s \), it indicates that service can simulate any state node in service', namely all state nodes are deleted in evolution from service to service'.

- If \( \forall p \) and \( sFs' \wedge sFs \) (it is marked as \( s \not\equiv s' \)), namely \( s \) is not weak equivalence to \( s' \), it indicates that service does not include the bisimulation state node of \( q \), namely they change. Old state nodes are changed in evolution from service to service'.

Shown as the Figure 4, the supply service evolves from service to service'. \( s_{service} \) and \( s_{service}' \) indicate the state set of service and service'. \( s-service = \{s_1, s_2, s_3, s_4 \} \) and \( s-service' = \{s_1, s_2, s_3, s_4, s_5, s_6 \} \). The business flow inside service includes \( \{s_1, s_2, s_3, s_4 \} \) and \( \{s_5, s_6 \} \). After evolution, the new business flow \( \{s_1, s_3 \} \) and \( \{s_5 \} \) are added to the old business of service, it indicates that the service adds its value when single service evolves to service'.

![Figure 4. Evolution of single service to service'](image3)
state nodes into one state node, including continued consolidation and parallel consolidation. The decomposition indicates to divide one state node to multiple state nodes, including continued decomposition and parallel decomposition. The combined actions are composed of elementary actions. E.g. Decomposing one state node to multiple state nodes is regarded as deletion and addition of state nodes. Consolidating one or multiple state nodes to other state nodes is regarded as deletion and change. The rule 3 gives determination method of state nodes to other state nodes is regarded as deletion and change. The rule 3 gives determination method from the view of the state node evolution in the global node of evolution of single service.

Rule 3 (determination of combined action type evolution of single service). service’ is the service after service evolution. \( p_i, \ldots, p_j \) is the state node in service. \( s_{p_i}, \ldots, s_{p_j} \) is the state set of \( p_i, \ldots, p_j \). \( p' \) is the state node of service’. \( s_{p_i'}, \ldots, s_{p_j'} \) is the state set of \( p_i', \ldots, p_j' \).

The determination rule of combined action type evolution of single service is described as follows:

- For \( s \in s_{p_i} \cup \ldots \cup s_{p_j}, s' \in s_{p_i'}, \) if \( \forall s, \exists s' \) and \( sSs' \land s'Ss \), it indicates that multiple state nodes in service are consolidated to one state node in service’, which belongs to parallel consolidation.

- For \( s \in s_{p_i}, s' \in s_{p_i}, \) if \( \forall s, \exists s' \) and \( sSs' \land s'Ss \), it indicates that one state node in service is decomposed to multiple state nodes in service’, which belongs to parallel decomposition.

- For \( s \in s_{p_i}, s' \in s_{p_i}, \) if \( \forall s, \exists s' \) and \( sSs' \land s'Ss \), it indicates to consolidate multiple state nodes in service to one state node in service’, which belongs to continued consolidation.

- For \( s \in s_{p_i}, s' \in s_{p_i}, \) if \( \forall s, \exists s' \) and \( sSs' \land s'Ss \), it indicates to decompose one state node in service to multiple state nodes in service’, which belongs to continued decomposition.

Shown as the Figure 5, the service service, service’ and service* have evolution association. 
\[
S_{\text{service}} = \{s_{21}, s_{22}, s_{23}, s_{24}, s_{25}, s_{26}\},
\]
\[
S_{\text{service}'} = \{s_{21}, s_{22}, s_{23}, s_{24}, s_{25}, s_{26}\},
\]
\[
S_{\text{service}''} = \{s_{21}, s_{22}, s_{23}, s_{24}, s_{25}, s_{26}\}.
\]

The evolution type is shown as the TABLE 2.

![Figure 5. Evolution of combined action type of single service](image)

TABLE 2. MEETING CONDITIONS AND TYPE OF COMBINED ACTION EVOLUTION OF SINGLE SERVICE

<table>
<thead>
<tr>
<th>Evolution service</th>
<th>Prior to evolution</th>
<th>After evolution</th>
<th>Met condition</th>
<th>Type of evolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td></td>
<td>service’</td>
<td>( sSs' \land s'Ss ) if ( s' \in s_{r_{12}} ), ( s \in s_{r_{12}} ) and ( \forall s, \exists s' ), then ( sSs' \land s'Ss )</td>
<td>continued decomposition</td>
</tr>
<tr>
<td>service’</td>
<td></td>
<td>service*</td>
<td>( sSs' \land s'Ss ) if ( s' \in s_{r_{12}} \cup s_{r_{12}}, s \in s_{r_{12}} ) and ( \forall s, \exists s' ), then ( sSs' \land s'Ss )</td>
<td>parallel decomposition</td>
</tr>
<tr>
<td>service’</td>
<td></td>
<td>service</td>
<td>( sSs' \land s'Ss ) if ( s \in s_{r_{12}}, s' \in s_{r_{12}} ) and ( \forall s', \exists s ), then ( sSs' \land s'Ss )</td>
<td>continued consolidation</td>
</tr>
<tr>
<td>service’</td>
<td></td>
<td>service</td>
<td>( sSs' \land s'Ss ) if ( s \in s_{r_{12}} \cup s_{r_{12}}, s' \in s_{r_{12}} ) and ( \forall s', \exists s ), then ( sSs' \land s'Ss )</td>
<td>parallel consolidation</td>
</tr>
</tbody>
</table>

C. Determination of Service Composition Evolution Type

Determine the type of the service combined evolution from the view of the state node evolution in the global business flow according to the definition of the strong simulation, weak simulation strong bisimulation and weak bisimulation in the bisimulation theory. The property 1 gives the relation between the strong simulation and weak simulation, between non-weak simulation and non-strong simulation and between non-weak bisimulation and non-strong bisimulation. The rule 4 gives the determination rule of the service composition
evolution type.

**Property 1.** The strong simulation must be weak simulation. Strong bisimulation must be weak bisimulation. Non weak simulation must be non strong simulation. Non weak bisimulation must be non-strong bisimulation.

**Proof:** according to the definition 5, if \( s \sim s' \), if \( s \xrightarrow{a} s'' \), \( s'' \) will meet \( s \xrightarrow{a} s'' \) and \( s' \sim s'' \), the definition 7 is met, namely \( s \Rightarrow s'' \) and \( \alpha \) is invisible action inside \( \alpha \), it is equivalent to 0 external action. If \( \alpha \) is an external visible action, it is equivalent to one external action, so the strong simulation must be weak simulation. The proving process is completed.

Similarly, it is proved that the strong bisimulation must be weak bisimulation. The non-weak simulation must be non-strong simulation. The non-weak bisimulation must be non-strong bisimulation.

**Rule 4 (determination of service composition evolution type).**

\[
\text{Service} = \{ \text{service}_1, \text{service}_2, \text{service}_3 \} \text{ is the service set. service is the originated evolution service. service' } \in \text{Service} \text{. service' is the evolved service. If the state transition model of service and service' are } (\text{state}, \text{relation}) \text{ and } (\text{state'}, \text{relation'}) \text{, for } \forall s \in \text{state}, 3s' \in \text{state'}, \text{ the following determination rule is met.}
\]

- If \( (sSs' \land s'Ss') \land (s\mathcal{R}s' \lor s'\mathcal{R}s) \) is met, namely \( s \) is equivalent to \( s' \) and \( s \) is not strong simulation of \( s' \) or \( s' \) is not strong simulation of \( s \), it indicates that service evolves to service' due to the change of the behaviors inside the service. The external observational behaviors do not change. The old service choreography protocol will keep valid. The evolution does not involve other copartner services. This evolution process is called the internal evolution or evolution at the service orchestration level.

- If \( s\mathcal{R}s' \land s'\mathcal{R}s \) is met, namely \( s' \) is the strong simulation of \( s \) and \( s \) is non-weak simulation of \( s' \). It indicates the service adds its value without change of the old business flow when service evolves to service'. Old choreography should be adjusted. The evolution involves other copartner services. This evolution process is called as the value-added evolution.

- If \( s\mathcal{R}s' \land s'\mathcal{R}s \) is met, namely \( s \) is strong simulation of \( s' \) and \( s \) is non-weak simulation of \( s' \), it indicates that the old business flow is reduced without new service functions in case of evolution from service to service'. The evolution involves the service choreography protocol, but the old service choreography can still be used. The evolution involves other copartner service. This evolution process is called as the reduced evolution of the service.

- If \( s\mathcal{R}s' \land s'\mathcal{R}s \) is met (it is marked as \( s \not\equiv s' \)), namely \( s \) is not weak equivalence to \( s' \). It indicates that the behaviors inside the service and observational behaviors outside the service will change in case of evolution from service to service'. The service evolution involves other copartner service in the service composition. This evolution process is called as the global evolution of the service.

The TABLE 3 gives four service composition evolution type and compares the influences of the service choreography protocol, evolution service origin and evolution service participation flow. The international evolution occurs inside single service and belongs to evolution at the service orchestration level. The value-added evolution, reduced evolution and global evolution occur among multiple services. The evolution involves the service choreography protocol and belongs to evolution at the service choreography level.

**TABLE 3. TYPE AND INFLUENCE OF SERVICE EVOLUTION BEHAVIOR**

<table>
<thead>
<tr>
<th>Type of service composition evolution</th>
<th>Choreography protocol change</th>
<th>Business flow</th>
<th>Evolution service origin</th>
<th>Evolution service participation</th>
</tr>
</thead>
<tbody>
<tr>
<td>internal evolution</td>
<td>invariable</td>
<td>change</td>
<td>unchanged</td>
<td></td>
</tr>
<tr>
<td>value-added evolution</td>
<td>change</td>
<td>add</td>
<td>add</td>
<td></td>
</tr>
<tr>
<td>reduced evolution</td>
<td>available</td>
<td>reduce</td>
<td>reduce</td>
<td></td>
</tr>
<tr>
<td>global evolution</td>
<td>change</td>
<td>change</td>
<td>change</td>
<td></td>
</tr>
</tbody>
</table>

Taking the service composition \( C_{w3} \) as one example, \( C_{w3} = \langle \text{Service, Bp, p} \rangle \), \( \text{Service} = \{ \text{service}_1, \text{service}_2, \text{service}_3 \} \), Service is the service set. Bp is the business flow set. p is the service choreography protocol. There are three business flows prior to \( C_{w3} \) evolution (shown as the Figure 6), including \( b_1 = (s_{11}, s_{12}, s_{21}, s_{22}, s_{13}) \), \( b_2 = (s_{11}, s_{12}, s_{21}, s_{22}, s_{23}, s_{24}, s_{31}, s_{32}, s_{33}, s_{34}, s_{26}) \) and \( b_3 = (s_{11}, s_{12}, s_{21}, s_{22}, s_{23}, s_{24}, s_{31}, s_{32}, s_{33}, s_{14}, s_{26}) \). The Figure 7-10 show four service evolution type of the service composition.
Internal evolution: the Figure 3 shows the internal business process evolution of the service originator. The evolution process is performed inside Service\(_1\). The business flow \(b_1\) evolves to \(b'_1 = (s_{11}, s_{12}, s_{21}, s_{22}, s_{23}, s_{24}, s_{31}, s_{32}, s_{34}, s_{35})\), but the external observational behaviors (interaction actions) do not change. This evolution does not affect the service choreography protocol and copartner service. The evolution can occur without the need of information the service participants.

Value-added evolution: under the premise of no change of old business flow, a new business flow is added and the observational behaviors (interaction behaviors) outside the service changes. Shown as the Figure 4, \(b_1, b_2, b_3\) does not change \(b_4 = (s_{11}, s_{12}, s_{21}, s_{22}, s_{23}, s_{24}, s_{31}, s_{32}, s_{34}, s_{35})\) and \(b_5 = (s_{11}, s_{12}, s_{21}, s_{22}, s_{23}, s_{24}, s_{31}, s_{32}, s_{33}, s_{36})\) are added in the evolved business flow. This evolution process involves the service choreography protocol and copartner service, but the evolution process may request to update the function nodes in other business flow such as \(s_{22}\) and \(s_{34}\), but the old business flow does not change prior to and after evolution.

Reduced evolution: quantity and functions of the old business flow reduce. The external observational behaviors of the service (interaction behavior) change. Shown as the Figure 5, \(b_1\) is deleted and \(b_2\) and \(b_3\) are reserved. The business nodes of \(b_3\) are included in service\(_1, service\(_2, and service\(_3, so the evolution affects other copartner services, compared to old service choreography protocol \(p\), the evolved service choreography protocol \(p'\) meets \(p' \subseteq p\), namely \(p\) includes the evolved business flow \(b_2\) and \(b_3\) choreography protocol, so the evolved business flow can operate under \(p\).

Global evolution: the external business and external observational behaviors (interaction behavior) of the service change, as shown the Figure 6. The business flow includes \(b_1 = (s_{11}, s_{12}, s_{21}, s_{22}, s_{15})\), \(b_2 = (s_{11}, s_{12}, s_{21}, s_{22}, s_{23}, s_{24}, s_{31}, s_{32}, s_{34}, s_{35}, s_{36})\) and \(b_3 = (s_{11}, s_{12}, s_{21}, s_{22}, s_{23}, s_{24}, s_{31}, s_{32}, s_{34}, s_{36})\). The node \(s_{23}\) is removed and the node \(s_{27}\) and \(s_{28}\) are added.

### VI. Example Analysis

The following example describes the determination problem of the service composition evolution type. The Figure 11 describes the order management service composition \(C_{WS}\) system of the high-end artworks sold via Internet. This system is composed of customer agent, supply service, logistics service and bank service system. Service = {service\(_x\), service\(_y\), service\(_z\), service\(_w\)} is the service set of \(C_{WS}\) and \(Bp = \{b_1, b_2, b_3\}\) is the business flow set of \(C_{WS}\). \(b_1 = (s_{11}, s_{12}, s_{13}, s_{41}, s_{42})\), \(b_2 = (s_{11}, s_{12}, s_{13}, (s_{41}, s_{42}) \| s_{21}), s_{22}, s_{23}, s_{24}\),
\[ s_{31}, s_{32}, s_{25}, s_{15} \] and \( b_3 = (s_{11}, s_{12}, s_{13}, (s_{41}, s_{42}, s_{21}, s_{22}, s_{23}, s_{24}, s_{25}, s_{16}) \) is executed in parallel.

Figure 11. Business flow of order management service prior to evolution

The supply service \( service_s \) aims to expand the sale scale as much as possible, under the premise of sale risk reduction, to abstract more consumers, based on the old pattern of first network payment and then delivery transaction, receiving and payment pattern based on the consumer trust check is introduced, so the supply service transaction, receiving and payment pattern based on the old scale as much as possible. under the premise of sale risk added sale mode and the credit function and fund return.

A. Determination of State Node Evolution Type

Taking the supply service \( service_s \) as one example, the state set of \( service_s \) is \( S_{service_s} = \{ s_{21}, s_{22}, s_{23}, s_{24}, s_{25} \} \) prior to evolution. After evolution, the state set of \( service_s \) is \( S'_{service_s} = \{ s_{21}, s_{22}, s_{23}, s_{24}, s_{25}, s_{26}, s_{27} \} \), \( s_{26}, s_{27} \) is the added state node after evolution. According to the determination rule, \( s_{22}, s_{24}, s_{25} \) is equivalent in observation prior to and after evolution, \( s_{21}, s_{23} \) belongs to value-added evolution.

B. Determination of Single Service Evolution Type

Taking \( service \) as one example, the evolution type is determined by the business flow change inside the \( service \). Prior to \( service \) evolution, the internal business flow includes \( b_1 = (s_{32}, s_{22}, s_{23}, s_{24}, s_{25}) \) and \( b_2 = (s_{25}) \). After evolution, the business flow includes \( b'_1 = (s_{32}, s_{22}, s_{23}, s_{24}, s_{25}) \), \( b'_2 = (s_{25}) \), \( b'_3 = (s_{32}, s_{26}, s_{23}, s_{24}, s_{25}) \), \( b'_{25} = (s_{25}) \) and \( b'_4 = (s_{21}, s_{26}) \). \( b'_{20} = (s_{21}, s_{22}, s_{23}, s_{24}) \), \( b'_{20} = (s_{21}, s_{22}, s_{23}, s_{24}) \) and \( b'_{20} = (s_{21}) \) the business flow does not include the corresponding business flow of \( b'_{20} \) prior to \( service \) evolution. For \( \forall s \in state \) and \( \exists s' \in state' \), \( s_{state} \wedge s_{state'} \) is met, namely \( s' \) is strong simulation of \( s \) and \( s \) is not strong simulation of \( s' \), \( service \) originates the value-added evolution. It indicates that evolution from \( service \) to \( service' \) adds the value without change of the old business flow. The old choreography protocol should be adjusted. The evolution involves other copartner services.

C. Determination of Service Composition Evolution Type

The participation service set and business flow set of the evolved service composition \( C'_{WSC} \) are \( Service'_c \) and \( Bp' \), the business flows of \( Service' = \{ service_1, service_2, service_3, service_4, service_5 \} \) and \( Bp' = \{ b'_1, b'_2, b'_3, b'_4, b'_5 \} \) include:

\( b'_1 = (s_{11}, s_{12}, s_{13}, s_{41}, s_{14}) \), \( b'_2 = (s_{11}, s_{12}, s_{13}) \), \( b'_3 = (s_{11}, s_{12}, s_{13}, s_{41}, s_{14}, s_{12}, s_{21}, s_{22}, s_{23}, s_{24}, s_{25}, s_{16}) \), \( b'_4 = (s_{11}, s_{12}, s_{13}, s_{21}, s_{26}, s_{23}, s_{24}, s_{25}, s_{16}) \), \( b'_5 = (s_{11}, s_{12}, s_{13}, s_{21}, s_{26}, s_{23}, s_{24}, s_{25}, s_{16}) \).

To compare \( C_{WSC} \) and \( C'_{WSC} \) prior to and after service composition evolution, \( b_1 = b'_1 \), \( b_2 = b'_2 \) and \( b_3 = b'_3 \) between the business flow \( Bp \) and \( Bp' \). The business flow \( b'_1 \), \( b'_2 \), \( b'_3 \) and \( b'_6 \) are added to \( Bp' \). \( STATE \) is the global state set of \( C_{WSC} \) \( \cdot \) \( STATE' \) is the global state set of \( C'_{WSC} \). \( STATE' \) is the global state set of \( C'_{WSC} \). For \( \forall s \in STATE \), \( \exists s' \in STATE' \) and \( s_{state} \wedge s_{state'} \) is met, namely \( s' \) is the strong simulation of \( s \) and \( s \) is not strong simulation of \( s' \). \( C'_{WSC} \) generates value-added evolution. It indicates that evolution from \( C_{WSC} \) to \( C'_{WSC} \) adds the service value without change of old business flow. The old choreography protocol should be adjusted. The evolution involves other copartner service.
VII. CONCLUSIONS

Determining the service composition evolution type can identify the logic relation among state nodes after evolution, reduce logic confliction, reduce the business flow dead lock and error and effectively determine the change of the service composition prior to and after evolution. The evolution implementation scheme can be identified by the evolution type. This study first proposes the close relation between implementation of service composition evolution and type of service composition evolution, determines the evolution type at single state node, single service and service composition and gives the determination rule. Determining evolution type of single state node can check whether the external behaviors of the nodes change. Determining type of single service evolution can get the quantity and logic relation of state nodes inside the service and change of the business flow. Determining the type of service evolution lays foundation for implementation scheme of evolution.

For determination of evolution type of Web service composition business flow evolution, this study mainly solves the following problems by using bisimulation theory, shown as the Figure 1. (a) Determine evolution type of single state node. The evolution type of single state node is divided into the observation equivalence, value-added, reduction and change via determination rule of weak simulation and observation equivalence. (b) Determine evolution type of single service: single service evolution type is divided into the elementary action type and combined action type. The elementary action includes addition, deletion and change of the state node. The service action type includes the continued consolidation, parallel consolidation, continued decomposition and parallel decomposition; (c) Determine the evolution type of service composition: the evolution type of service composition is divided into the internal type, value-added type, reduced type and global evolution type from the view of the business flow according to the change of business flow inside the participating services by using strong simulation and weak simulation determination rule.

ACKNOWLEDGMENT

The work described in this study was supported by the National Natural Science Foundation, China(No.61272125), Research Fund for the Doctoral Program of Higher Education of China, China (No.20121333110014), the Natural Science Foundation of Hebei Province, China(No.F2011203234), and Key Technology Support Program of Hebei Province, China (No.11213562).

REFERENCES


Dianloing You was born in Inner Mongolia Province, China in 1981. He is currently a Ph.D. Candidate in the School of Information Science and Engineering, Yanshan University, Qinhuangdao, China. His research interests include service computing and service evolution

Limin Shen was born in Heilongjiang Province, China in 1962. He is currently a professor in the School of Information Science and Engineering, Yanshan University, Qinhuangdao, China. His research domains service-oriented computing and system Integration and information.