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 offered 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 is evolving from single-function and small-granularity atomic service to the big-granularity service based on the complicated business flow and component integration. The service composition evolution is not only the atomic service replacement under the small-granularity service composition [2]. The Web service composition at the service orchestration and service choreography level is described in the industry. The service orchestration describes the business flow execution inside the service and between services. The 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

© 2014 ACADEMY PUBLISHER
doi:10.4304/jsw.9.6.1444-1452

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.

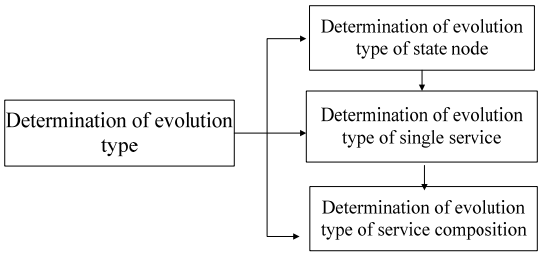

Figure 1. Determination of evolution type

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 $s_i \xrightarrow{a_{i+1}} s_{i+1} \xrightarrow{a_{i+2}}, \cdots, \xrightarrow{a_j} s_j$, it is marked as $(s_i, s_{i+1}, \cdots, s_j)$ or $(s_i \triangleright s_j)$, $fn(b)$ indicates the set of the state in b. Generally s_0 indicates initial state. s_f indicates the ending state. For the state sequence $(s_i, s_{i+1}, \cdots, s_j)$, $s_0 = s_i$ and $s_f = s_j$. Multiple business flows of the service composition may be available. They are named as b_1, b_2, …, b_n in turn. The state node of single business flow b_i of the service composition may distribute inside the service. If $service_s$ 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 $service_s$ .multiple such internal business flows may exist in $service_s$. k th

© 2014 ACADEMY PUBLISHER

internal business flow is marked as $b_{i(s):k}$. $b_{i(s):k}$ indicates k^{th} internal business flow of the business flow b_i in the service $service_s$.

Definition 3 (WSC of Web service composition). it is a two tuple $WSC =< Service, Bp, p >$, wherein:

- $Service = \{ service_1, service_2, \cdots, service_n \}$ 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.
- $Bp = \{ b_1, b_2, \cdots, b_m \}$ 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 *(state,relation)* is a state transition system. If $s_p, s_{p'}, s_q, s_{q'} \in state$ and $\Re = \{(s_p, s_{p'}), (s_q, s_{q'}), \cdots\}$ is two tuple relation in $state$. For any $s_p \Re s_q$, 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 $s_p' \Re s_q'$, $\Re$ is called as a strong simulation in *(state,relation)*. If a strong simulation $\Re$ makes $s_p \Re s_q$, s_q is called as strong simulation s_p.

Definition 5 (strong bisimulation and strong equivalence) . if two duple relation $\Re$ and its reverse $\Re^{-1}$ is the strong simulation of the state transition system *(state,relation)*, $s_p, s_q \in state$ and $\Re = \{(s_p, s_{p'}), (s_q, s_{q'}), \cdots\}$ is two duple relation in $state$. If a strong simulation $\Re$ makes $s_p \Re s_q$, two state s_p and s_q are strong bisimulation or strong equivalence. It is marked as $s_p \sim s_q$.

Definition 6 (weak simulation) . assume that *(state,relation)* is a state transition system, $s_p, s_{p'}, s_q, s_{q'} \in state$ and $S = \{(s_p, s_{p'}), (s_q, s_{q'}), \cdots\}$ is two tuple relation in $state$. For any $s_p S s_q$, the following condition is met:

If $s_p \overset{e}{\Rightarrow} s_{p'}$, $s_{q'}$ makes $s_q \overset{e}{\Rightarrow} s_{q'}$ and $s_p' S s_q'$, S is called as one weak simulation in *(state,relation)*. If one weak simulation S makes $s_p S s_q$, s_q is the weak simulation of s_p. e indicates visible action sequence $\lambda_1 \cdots \lambda_n$. $s_p \overset{e}{\Rightarrow} s_{p'}$ indicates $s_p \Rightarrow \xrightarrow{\lambda_1} s_{p1} \cdots \Rightarrow \xrightarrow{\lambda_n} s_{pn} \Rightarrow s_{p'}$. $\Rightarrow$ indicates zero or multiple internal actions.

Definition 7 (weak bisimulation, weak equivalence or observational equivalence) . if two tuple relation S and its reverse S^{-1} is the weak simulation of the state transition system *(state,relation)*. $s_p, s_q \in state$, $S = \{(s_p, s_{p'}), (s_q, s_{q'}), \cdots\}$ is a two-tuple relation in $state$. If one weak simulation S makes $s_p S s_q$, the state s_p and s_q are called as weak bisimulation, weak equivalence or observational equivalence and it is marked as $s_p \approx s_q$.

The bisimulation equivalence is the common equivalence relation in the process algebra and is divided into the strong bisimulation equivalence and weak bisimulation equivalence by internal action ignorance or not. The strong bisimulation regards all actions (including invisible actions inside) equally. The weak bisimulation 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 [6,9-10], 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 bisimulation in the bisimulation 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 $sSs' \wedge s'Ss$ (it is also marked as $s \approx s'$), 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 $sSs' \wedge s' \not S s$, namely s' is weak simulation of s and s is not weak simulation of s', it indicates that p' includes all observational

© 2014 ACADEMY PUBLISHER

behaviors of p and p does not include the state transition, namely the observational behaviors **adds value** when p evolves to p'.

- If $s'Ss \wedge s\not S s'$, 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 $s\not S s' \wedge s'\not S s$ (it is also called as $s \not\approx s'$), 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 Figure3. if the state transition sequence is $\cdots s_i \xrightarrow{a_{i+1}} s_{i+1} \xrightarrow{a_{i+2}} \cdots \xrightarrow{a_j} s_j \cdots$ prior to evolution, the TABLE 1 gives the mapping of the state transition sequence and evolution type after evolution.

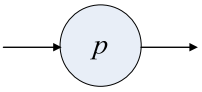

Figure 2. State node prior to evolution

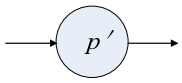

Figure 3. State node after evolution

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

state transition sequence after the state node p evolves to p'	Evolution type
$\cdots s_i \xrightarrow{a_{i+1}} s_{i+1} \xrightarrow{a_{i+2}} \cdots \xrightarrow{a_j}$	observational equivalence
$\cdots s_i \xrightarrow{a_{i+1}} s_{i+1} \xrightarrow{a_{i+2}} \cdots \xrightarrow{a_k} s_k \cdots \xrightarrow{a_j} s_j \cdots$	added
$\cdots s_i \xrightarrow{a_{i+1}} s_{i+1} \xrightarrow{a_{i+2}} \cdots$	reduced
$\cdots s_i \xrightarrow{a_{i+1}} s_{i+1} \xrightarrow{a_{i+2}} \cdots \xrightarrow{a_j} s_k \cdots$	changed

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* . q is the state node in the *service'* after evolution. $s_q^{'}$ is the state set of q . *state* and *state'* are the state set of *service* and *service'* . $s_p \subseteq state$, $s_q^{'} \subseteq state'$. The determination rule is described as follows: for $\forall s' \in s_q, \forall s \in state$,

- If $\exists q$ and $s\not S s' \wedge s'\not S s$, 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 $s'\Re s \wedge s\not S s'$, 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 $s\not S s' \wedge s'\not S s$ (it is marked as $s \not\approx s'$), 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 $\{ s_2, s_3, s_4 \}$. After evolution, the new business flow $\{ s_1, s_5 \}$ and $\{ s_6 \}$ are added to the old business of *service* , it indicates that the service adds its value when single *service* evolves to *service'* .

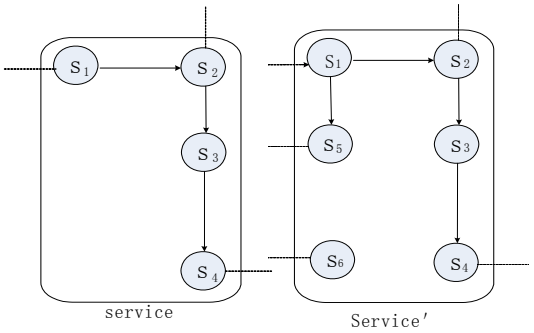

Figure 4. Evolution of single *service* to *service'*

The combined actions of single service evolution include consolidation and decomposition. The consolidation indicates to consolidate two or multiple

© 2014 ACADEMY PUBLISHER

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 combined action type evolution of single service.

Rule 3 (determination of combined action type evolution of single service) . *service′* is the service after *service* evolution. $p_i,\cdots,p_j$ is the state node in *service*. $s_{p_i},\cdots,s_{p_j}$ is the state set of $p_i,\cdots,p_j$. p' is the state node of *service′*. $s_{p_i'},\cdots,s_{p_j'}$ is the state set of $p_i',\cdots,p_j'$. The determination rule of combined action type evolution of single service is described as follows:

- For $s\in s_{p_i}\bigcup\cdots\bigcup s_{p_j}, s'\in s_{p'}$, if $\forall s,\exists s'$ and $s\mathrm{S}s'\wedge s'\mathrm{S}s$, it indicates that multiple state nodes in *service* are consolidated to one state node in *service′*, which belongs to **parallel consolidation.**
- For $s\in s_p, s'\in s_{p_i'}\bigcup\cdots\bigcup s_{p_j'}$, if $\forall s,\exists s'$ and $s\mathrm{S}s'\wedge s'\mathrm{S}s$, it indicates that one state node in *service* is decomposed to multiple state nodes in *service′*, which belongs to **parallel decomposition.**
- For $s\in s_{p_j}, s'\in s_p$, if $\forall s',\exists s$ and $s\mathrm{S}s'\wedge s'\mathrm{S}s$, it indicates to consolidate multiple state nodes in *service* to one state node in *service′*, which belongs to **continued consolidation.**
- For $s\in s_p, s'\in s_{p_j'}$, if $\forall s,\exists s'$ and $s\mathrm{S}s'\wedge s'\mathrm{S}s$, 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. $S_{service}=\{s_{21},s_{22},s_{23},s_{24},s_{25},s_{26}\}$, $S_{service'}=\{s_{21},s_{22},s_{23a},s_{23b},s_{24},s_{25},s_{26}\}$ and $S_{service''}=\{s_{21},s_{22a},s_{22b},s_{23},s_{24},s_{25},s_{26}\}$. The evolution type is shown as the TABLE 2.

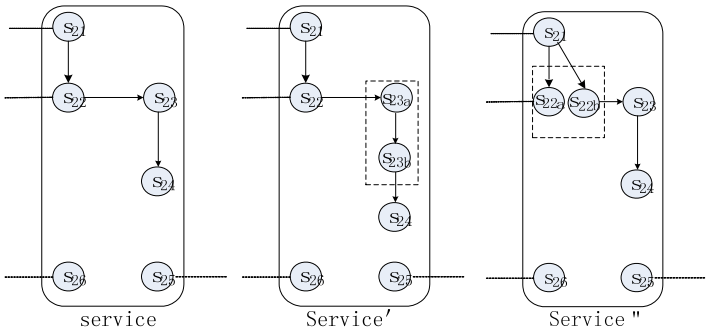

Figure 5. Evolution of combined action type of single service

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

Evolution service		Met condition	Type of evolution
Prior to evolution	After evolution		
service	*service′*	$s\mathrm{S}s'\wedge s'\mathrm{S}s$ If $s'\in s_{s_{23b}}, s\in s_{s_{23}}$ and $\forall s,\exists s'$, then $s\mathrm{S}s'\wedge s'\mathrm{S}s$	continued decomposition
	service″	If $s'\in s_{s_{23a}}\bigcup s_{s_{23b}}, s\in s_{s_{23}}$ and $\forall s,\exists s'$, then $s\mathrm{S}s'\wedge s'\mathrm{S}s$	parallel decomposition
service′	*service*	If $s\in s_{s_{23b}}, s'\in s_{s_{23}}$ and $\forall s',\exists s$, then $s\mathrm{S}s'\wedge s'\mathrm{S}s$	continued consolidation
service″		If $s\in s_{s_{23a}}\bigcup s_{s_{23b}}, s'\in s_{s_{23}}$ and $\forall s',\exists s$, then $s\mathrm{S}s'\wedge s'\mathrm{S}s$	parallel consolidation

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

© 2014 ACADEMY PUBLISHER

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 $s_p \Re s_q$, if $s_p \xrightarrow{\alpha} s_p^{'}$, $s_q^{'}$ will meet $s_q \xrightarrow{\alpha} s_q^{'}$ and $s_p^{'} \Re s_q^{'}$, the definition 7 is met, namely $s_p \Rightarrow s_p^{'}$, so If $s_q^{'}$ exists to make $s_q \Rightarrow s_q^{'}$ and α is invisible action inside α, it is equivalent to 0 external action. If α 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). $Service = \{ service_1, service_2, \cdots, service_n \}$ is the service set. $service$ is the originated evolution service. $service \in Service$. $service'$ is the evolved $service$. If the state transition model of $service$ and $service'$ are $(state, relation)$ and $(state', relation')$, for $\forall s \in state, \exists s' \in state'$, the following determination rule is met.

- If $(s \mathrm{S} s' \wedge s' \mathrm{S} s) \wedge (s \not\Re s' \vee s' \not\Re s)$ 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 as the **internal evolution** or **evolution at the service orchestration level**.
- If $s \Re s' \wedge s' \not\Re s$ 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 $s' \Re s \wedge s \not\Re s'$ 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 $s \not\Re s' \wedge s' \not\Re s$ is met (it is marked as $s \not\approx s'$), 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 origination 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

Type of service composition evolution	Choreography protocol change	Business flow	
		Evolution service origination	Evolution service participation
internal evolution	invariable	change	unchanged
value-added evolution	change	add	add
reduced evolution	available	reduce	reduce
global evolution	change	change	change

Taking the service composition C_{WS} as one example, $C_{WS} =< Service, Bp, p >$, $Service = \{ service_1, service_2, service_3 \}$, $Service$ is the service set. Bp is the business flow set. p is the service choreography protocol. There are three business flows prior to C_{WS} *evolution (*shown as the Figure 6), including $b_1 = (s_{11}, s_{12}, s_{21}, s_{22}, s_{13})$, $b_2 = (s_{11}, s_{12}, s_{21}, s_{22}, s_{23}, s_{24}, s_{31}, s_{32}, s_{15}, s_{26})$ and $b_3 = (s_{11}, s_{12}, s_{21}, s_{22}, s_{23}, s_{24}, s_{31}, s_{25}, s_{14}, s_{26})$ The Figure 7-10 show four service evolution type of the service composition.

© 2014 ACADEMY PUBLISHER

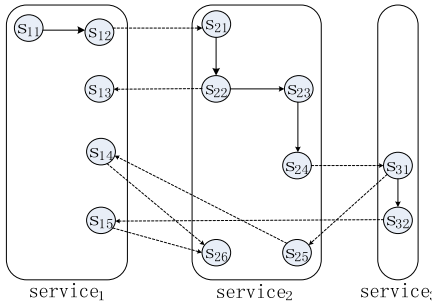

Figure 6. business flow of original service composition

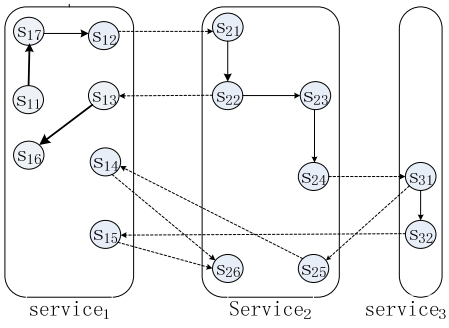

Figure 7. Business flow after internal evolution

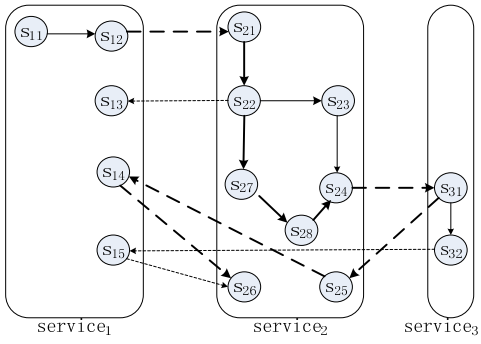

Figure 8. Business flow after value-added evolution

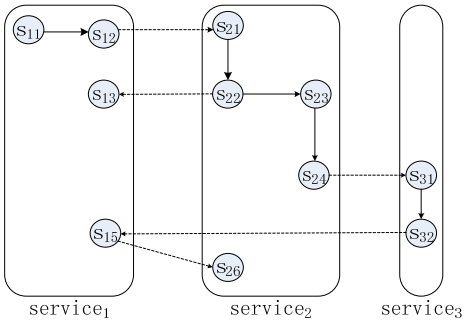

Figure 9. Business flow after reduced evolution

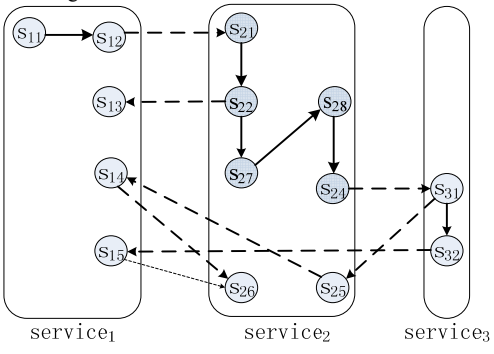

Figure 10. business flow after global evolution

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 $b_1^{'} = (s_{11}, s_{17}, s_{12}, s_{21}, s_{22}, s_{13}, s_{16})$, 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 $b_4 = (s_{11}, s_{12}, s_{21}, s_{22}, s_{27}, s_{28}, s_{24}, s_{31}, s_{25}, s_{14}, s_{26})$ and $b_5 = (s_{11}, s_{12}, s_{21}, s_{22}, s_{27}, s_{28}, s_{24}, s_{31}, s_{32}, s_{15}, s_{26})$ 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_{24} , 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_3 is deleted and b_1 and b_2 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_1 and b_2 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, shown as the Figure 6. The business flow includes $b_1 = (s_{11}, s_{12}, s_{21}, s_{22}, s_{13})$, $b_2^{'} = (s_{11}, s_{12}, s_{21}, s_{27}, s_{28}, s_{23}, s_{24}, s_{31}, s_{32}, s_{15}, s_{26})$ and $b_3^{'} = (s_{11}, s_{12}, s_{21}, s_{22}, s_{27}, s_{28}, s_{24}, s_{31}, s_{25}, s_{14}, s_{26})$ 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. $Service = \{service_a, service_s, service_l, service_b\}$ is the service set of C_{WS} and $Bp = \{b_1, b_2, b_3\}$ is the business flow set of C_{WS} . $b_1 = (s_{11}, s_{12}, s_{13}, s_{41}, s_{14})$, $b_2 = (s_{11}, s_{12}, s_{13}, ((s_{41}, s_{42}) \| s_{21}), s_{22}, s_{23}, s_{24},$

© 2014 ACADEMY PUBLISHER

$s_{31}, s_{32}, s_{25}, s_{15})$ and $b_3 = (s_{11}, s_{12}, s_{13}, ((s_{41}, s_{42}) \| s_{21}), s_{22}, s_{23}, s_{24}, s_{31}, s_{32}, s_{25}, s_{33}, s_{16}), \|$ is executed in parallel.

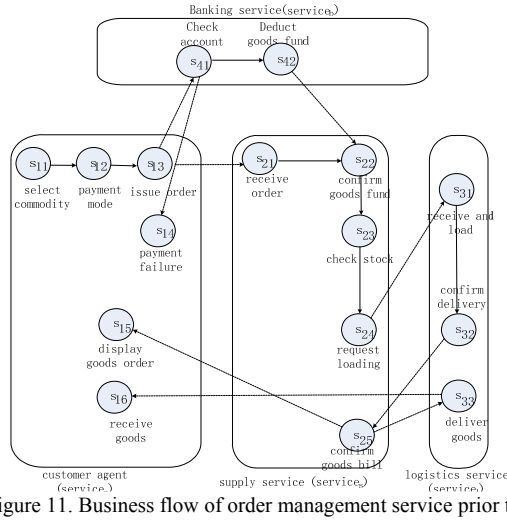

Figure 11. Business flow of order management service prior to evolution

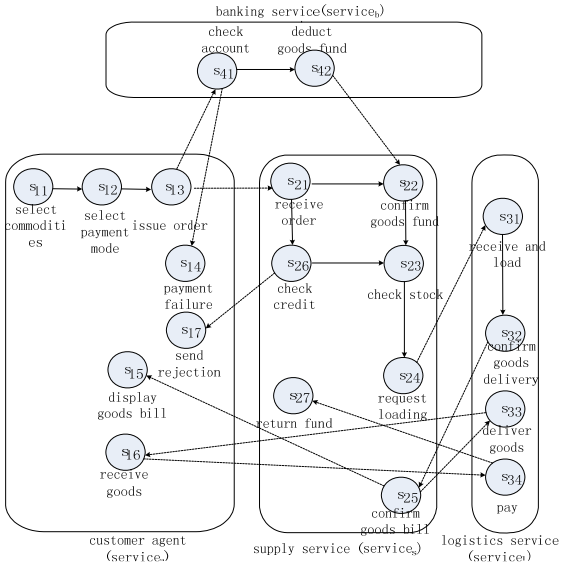

Figure 12. Business flow of order management service after evolution

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 ($service_s$) evolves to the $service_s'$ according to this added sale mode and the credit function and fund return function is added in the service, shown as the Figure 12.

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 $service_s$ as one example, the evolution type is determined by the business flow change inside the $service_s$. Prior to $service_s$ evolution, the internal business flow includes $b_{2(s):1} = ((s_{21}, s_{22}) \| s_{22}), s_{23}, s_{24})$ and $b_{2(s):2} = \{s_{25}\}$. After evolution, the business flow includes $b'_{2(s):1} = ((s_{21}, s_{22}) \| s_{22}), s_{23}, s_{24})$, $b'_{2(s):2} = \{s_{25}\}$, $b'_{2(s):3} = \{s_{21}, s_{26}, s_{23}, s_{24}\}$, $b'_{2(s):4} = \{s_{27}\}$ and $b'_{2(s):5} = \{s_{21}, s_{26}\}$. $b_{2(s):1} = b'_{2(s):1} = ((s_{21}, s_{22}) \| s_{22}), s_{23}, s_{24})$ and $b_{2(s):2} = b'_{2(s):2} = (s_{25})$, the business flow does not include the corresponding business flow of $b'_{2(s):3}$, $b'_{2(s):4}$ and $b'_{2(s):5}$ prior to $service_s$ evolution. For $\forall s \in state$ and $\exists s' \in state'$, $s \Re s' \wedge s' \not\Re s$ is met, namely s' is strong simulation of s and s is not strong simulation of s', $service_s$ originates the value-added evolution. It indicates that evolution from $service_s$ to $service_s'$ 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'_{WS} are $Service'$ and Bp', the business flows of $Service' = \{service_a, service_s, service_l, service_b\}$ and $Bp' = \{b'_1, b'_2, b'_3, b'_4, b'_5, b'_6\}$ include: $b'_1 = (s_{11}, s_{12}, s_{13}, s_{41}, s_{14})$, $b'_2 = (s_{11}, s_{12}, s_{13}, (s_{41}, s_{42}) || (s_{21}), s_{22}, s_{23}, s_{24}, s_{31}, s_{32}, s_{25}, s_{15})$, $b'_3 = (s_{11}, s_{12}, s_{13}, (s_{41}, s_{42}) || (s_{21}), s_{22}, s_{23}, s_{24}, s_{31}, s_{32}, s_{25}, s_{33}, s_{16})$, $b'_4 = (s_{11}, s_{12}, s_{13}, s_{21}, s_{26}, s_{17})$, $b'_5 = (s_{11}, s_{12}, s_{13}, s_{21}, s_{26}, s_{23}, s_{24}, s_{31}, s_{32}, s_{25}, s_{15})$, $b'_6 = (s_{11}, s_{12}, s_{13}, s_{21}, s_{26}, s_{23}, s_{24}, s_{31}, s_{32}, s_{25}, s_{33}, s_{16})$.

To compare C_{WS} and C'_{WS} 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'_4, b'_5 and b'_6 are added to Bp'. $STATE$ is the global state set of C_{WS}. $STATE'$ is the global state set of C'_{WS}. $STATE'$ is the global state set of C'_{WS}. For $\forall s \in STATE$, $\exists s' \in STATE'$ and $s \Re s' \wedge s' \not\Re s$ is met, namely s' is the strong simulation of s and s is not strong simulation of s'. C_{WS} originates value-added evolution. It indicates that evolution from C_{WS} to C'_{WS} adds the service value without change of old business flow. The old choreography protocol should be adjusted. The evolution involves other copartner service.

© 2014 ACADEMY PUBLISHER

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 implementation 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

[1] Song Wei. "Research on Dynamic Evolution of Web Service Composition", Institute of Computer Software of Nanjing University, 2010, pp.10.
[2] HU Hai Tao, LI Gang , HAN Yan Bo."An Approach to Business-User-Oriented Larger-Granularity Service Composition",Chinese Journal of Computers, vol.28,pp.694,2005.
[3] Song Wei, Ma Xiao-Xing, Lu Jian, "Instance Migration in Dynamic Evolution of Web Service Compositions", Chinese Journal of Computers, Vol.32,pp.1816-1831,September 2009.
[4] Peltz c, "Web services orchestration and choreography", IEEE Computer, vol.36,pp.46, 2003.
[5] Cambronero M. E, Díaz G, Valero V, Martínez E, "Validation and verification o f Web services choreographies by using timed automata",The Journal of Logic and Algebraic Programming,vol.80,pp.25-49,2011.
[6] Song Wei, Lu Jian, Ma Xiao-Xing, and Cao Chun, "A Method of Dynamic Evolution of Web Service Choreography", 2010 CCF National Conference on Service Computing,pp.274-286, 2010.
[7] RYU S H, CASATI F,SKOGSRUD H, "Supporting the Dynamic Evolution of Web Service Protocols in Service-Oriented Architecture,"ACM Transactions on the Web,vol.5,pp.1-43,November 2007.
[8] Paul C and Zhu Yaoling, "A Semantical Framework for the Orchestration and Choreography of Web Services," Electronic Notes in Theoretical Computer Science,vol.151,pp.3-18, 2006.
[9] SONG Wei,TANG JinHui1, ZHANG GongXuan,and MA XiaoXing, "Substitutability analysis of WS-BPEL services," Scientia Sinica(Informationis), vol.42.pp.264,2012.
[10] Deng Shui-guang, Huang Long-tao, and Yin Jian-wei, "Technical framework for Web Services composition and its progress". Computer Integrated Manufacturing Systems, vol.17, pp.405-411, February 2011.
[11] Liske N, Lohmann N, Stahi c, WolfK, "Another Approach to Service Instance Migration.Proceedings of the Joint International Conference on Service-Oriented Computing and ServiceWave".Stockholm,Sweden, pp.607-621, 2009.
[12] Papazoglou M P, "The challenges of service evolution", Proceedings of the International Conference on Advanced Information Systems Engineering, pp.1-15,2008.
[13] Guisheng FanAn, "Approach to Analyzing Time Constrained Service Composition",Journal Of Computers, vol.6.pp.1723-1731,Agust 2011.
[14] Minghui Wu, Xianghui Xiong, Jing Ying, Canghong Jin,and Chunyan Yu,"QoS-driven Global Optimization Approach for Large-scale Web Services Composition", Journal Of Computers, vol.6,pp.1452-1460,July 2011.
[15] Yuxiang Dong, "Modeling and Performance Evaluation of Service Choreography based on Stochastic Petri Net", Journal Of Computers, vol 5,pp.516-516,April 2010

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. He 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.

© 2014 ACADEMY PUBLISHER