Petri nets-based method to model and analyse the self-healing web service composition

Liqiong Chen
Department of Computer Science and Information Engineering, Shanghai Institute of Technology, Shanghai, China
and
Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai, China
Email: lqchen@sit.edu.cn

Guisheng Fan* and Huanhuan Zhang
Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai, China
Email: gsfan@ecust.edu.cn
Email: hzhang@ecust.edu.cn
*Corresponding author

Lizhong Xiao
Department of Computer Science and Information Engineering, Shanghai Institute of Technology, Shanghai, China
Email: xiao1z@sit.edu.cn

Abstract: Service composition is an important means for integrating the individual web services to create new value added systems that can satisfy complex requirements. Such applications are subject to unexpected failure owing to the distributive and heterogeneous environment of web service. In this paper, we propose a Petri nets-based method to model and analyse the self-healing service composition. Aspect oriented programming is used to extract composition processes and self-healing strategy as the core and crosscutting concerns. A self-healing strategy of component and service composition is proposed, and the related operations of self-healing are abstracted as meta-objects by using reflection mechanism. Petri nets are used to construct the base layer model, meta layer model, meta-object protocol and other components, thus forming the self-healing model. The operational semantics and related theories of Petri nets help prove its effectiveness and correctness. A simulation example shows that our approach correctly describes the self-healing process of service composition, and contributes to improving the quality.

Keywords: self-healing; service composition; aspect orientation; Petri nets; reflection.


Biographical notes: Liqiong Chen is an Associate Professor at the Department of Computer Science and Information Engineering Shanghai Institute of Technology. Her research interests include formal methods for complex software systems.

Guisheng Fan is an Associate Professor at the Department of Computer Science and Engineering, East China University of Science and Technology. His research interests include formal methods for complex software systems, service oriented computing, and techniques for analysis of software architecture.

Huanhuan Zhang is an Associate Professor at the Department of Computer Science and Engineering, East China University of Science and Technology. His research interests include software engineering and formal method.
1 Introduction

Service composition provides a mechanism for software integration, where different enterprise solutions cooperate to achieve a common goal, and primarily concerns the requests of users that cannot be satisfied by any available service, whereas a composite service obtained by combining a set of available web services might be used (Hui et al., 2013). With the increase of task service numbers, applications can be very complex in structure. How to improve the reliability of service composition has become an research problem.

Self-healing web services means embodying them with extra functionalities, which permit them for example to handle operation resumption after failure (Psaier and Dustdar, 2010). It is no easy task to realise it because of the following reasons: first, web services are more error prone to failures, if one of the participating services fails, the composition plan will not be able to select an alternative service to replace it. It may be forced to abort the composition or the composition may not behave as expected. Second, web services can be subject to the unexpected failures. It is difficult for the developers to plan recovery strategy at the design time. The failure of participants will lead to the cancelation of the whole process. Therefore, it requires that the composite service must have self-healing ability, which means that composite service can heal itself if any execution problem occurs, in order to complete its execution successfully and meet the users’ constraints.

This paper investigates how to model and analyse self-healing service composition based on user requirements. Below summarises our main contributions:

1 We propose a service composition net (SCN) to model different components of service composition. Then a self-healing strategy of component and service composition is proposed.

2 Aspect orientation (Kiczales et al., 1997) is used to extract the requirements of self-healing service composition. The functional requirement is abstracted as the core concern, while the related operation of self-healing is abstracted as meta-object by using reflection mechanism.

3 Computation tree logic (CTL) is used to convert the self-healing strategy into the CTL formulas. The related theories of Petri net and its state space help verify the effectiveness of self-healing strategy.

The remainder of this paper is organised as follows: Section 2 is the requirements specification. Section 3 is used to model self-healing service composition, Section 4 analyses the effectiveness of proposed method. Section 5 analyses the feasibility of proposed method by an example. Section 6 presents some related works and Section 7 is the conclusion.

2 Requirements specification

2.1 Requirements of service composition

First of all, we will give the requirements of service composition, which includes the set of components, the set of available services, the relationship between components, the mapping relationship between the component and available service, etc. Then, we use will an export service as a motivating example, which can explain the related definition and theorem throughout the paper. Finally, we will propose a self-healing strategy for components and service compositions.

Definition 1: The requirement of service composition is a six-tuple: \( \Xi = (C, WS, RL, TW, RT, DT) \):

1 \( C, WS \) are the set of task and available service.

2 \( RL : C \times C \rightarrow \{ >, +, \|, n \} \) is the relation function between components. \( >, +, \|, n \) are the sequence, choice, parallel and loop relation. If the relationship between \( C_i \) and \( C_j \) is sequence, then \( C_i \) is the forward component of \( C_j \), \( C_j \) is the afterward component of \( C_i \). For \( k(C_i) \), Back(\( C_i \)) are the forward and afterward set of \( C_i \).

3 \( TW : C \rightarrow WS^* \) is the available service of component. \( TW(C_i) = WS_i = \{ WS_{i,1}, WS_{i,2}, \ldots, WS_{i,n} \} \) is the available service of component \( C_i \).

4 \( RT : C \rightarrow N \times N^* \) is the attribute function of component. \( RT(C_i) = \{ e_t, c_{n_t} \}, e_t, c_{n_t} \) are the running time and checkpoints of component \( C_i \).

5 \( DT \) is the deadline of service composition, the time unit can be set according to the actual requirement.

The execution process of service composition is that component \( C_i \) will randomly select an available service \( WS_{i,j} \in WS_i \) to realise its function when the component gets the input parameters. We assume:
2.2 Motivating example

In order to illustrate our approach, we will use a motivating example which consists of a set of services described in the context of export service. First, the system will look up the relevant information and select the destination \((C_1)\). Packaging services \((C_2)\) and comodity inspection bureau inspection \((C_3)\) and insurance processing \((C_4)\), where export transport ordering includes water transport ordering \((C_5)\) and airport ordering \((C_6)\), the corresponding results will feedback to the exporter. All the information is confirmed by exporter and will be sent to the regulatory authorities \((C_7)\) for checking. Finally, financial service \((C_8)\) is used to check the related taxes and deadlines to each component and construct the invoked services of component.

Definition 3: Let \(\Xi\) be the requirement model of service composition, then the steps for computing the deadline and the invoked service are:

1. Computing and initialising the critical set of component \(KC\), \(KC = \emptyset\).
   
   a. The system will divide the business process into the sub business process \(SBP\) and \(C_i \in sbp\), where makes \(\forall C_j \in sbp\), there are \(RL(C_i, C_j) \notin \{\|, +\}\) and \(\exists C_k \in sbp\), which makes \(RL(C_i, C_k) \cup RL(C_k, C_i) = \Rightarrow\).

   b. \(\forall sbp \in SBP\), if \(C_i \in sbp\), which makes \(\forall C_j \in C\), there is \(RL(C_i, C_j) \notin \{\|, +\}\), then \(KC = KC \cup sbp\), \(\text{temp} = SBP - sbp\) and do Step c.

   c. The system will do Step b until no new sub business process is added to \(KC\), \(\forall sbp \in temp\). Computing the running time of \(sbp\):

   \[
   ext(sbp) = \sum_{C_j \in sbp} et_j \quad \text{and do Step d.}
   \]

   d. Adjusting the critical set of component, \(\forall sbp \in SBP\),

   \[
   tc(sbp) = \{sbp | RL(C_i, C_j) \in \{\|, +\}, \hfill \]

   \[
   C_j \in sbp, \quad C_k \in sbp \cup sbp, \hfill \]

   \[
   \text{if } ext(sbp) = \max\{tc(sbp)\}, \quad \text{where} \hfill \]

   \[
   sbp_k \in tc(sbp), \quad KC = KC \cup sbp_k, \hfill \]

   \[
   \text{temp} = temp - tc(sbp). \quad \text{If temp} \neq \emptyset, \text{then do Step d, otherwise, it will do Step (2).}
   \]

2. Filtering the set of available service:

   a. Allocating the static deadline \(sdt_i\) for component \(C_i\), \(\forall C_i \in KC\), \(sdt_i = DT - \sum_{C_j \in KC - C_i} et_j\),

   \[
   \forall C_j \in C, \text{ if } RL(C_i, C_j) \in \{\|, +\}, \text{ then } sdt_j = sdt_i.
   \]

   b. Filtering the set of available service according to the number of checkpoint and deadline, the upper deadline of required available service in component \(C_i\) is equal to: \([\frac{sdt_i - et_i}{cn_i} + 1]\). \([X]\) is the rounded down operation. \(sdt_i, et_i, cn_i\) are the static deadline, running time and the number of checkpoint in component \(C_i\).

3. Computing the dynamic deadline \(dt_i\) for component \(C_i\):

   a. \(\forall C_i \in KC\),

   \[
   dt_i = DT - T\text{Now} - \sum_{C_j \in KC - C_i \in EC} et_j.
   \]
First of all, we will process the requirement of service composition (Definition 1) by using the above steps, then compute the deadline and the required available services of each component.

3 Modelling self-healing service composition

3.1 Syntax and semantics

Petri net (PN) is a formal language for describing the concurrency system because its semantics are formally defined (Tadao, 1989). We will use Petri net to model service, each service needs to get the input parameters and output the results. First, we will introduce the related concepts.

Definition 4: A nine-tuple $\Omega = (P, N, IO, D, \Gamma, A_T, A_F, \lambda, C_t, M_0)$ is called service composition net (SCN) if:

1. $PN = (P, T, F)$ is a basic Petri net. $P, T, F$ represent the place, transition and arc.
2. $IO \subseteq P$ is a special type of place, which is the interface of $\Sigma$ and denoted by dotted circle. It is used to realise the interaction between services.
3. $D$ is the non-empty finite individuality set. $f_D, f_S$ are the predicate set and symbol set.
4. $\Gamma = \{\Gamma_i | i \in N\}$ is the finite set of SCN. Each element is also a page of $\Omega$.
5. $A_T : T \rightarrow f_D$, the free variable in $A_T(t)$ must be the free variable in the directed arc with one end of the arc is $t$.
6. $A_F : F \rightarrow f_S$, if $(p, t) \in F$ or $(t, p) \in F$, then $A_F(p, t)$ or $A_F(t, p)$ is the $n$-symbol set, the default value is empty.
7. $\lambda : T \rightarrow N$ is the priority of transition, the default value is 0. The smaller the value of $\lambda$, the greater the priority of transition.
8. $C_t : P \rightarrow N^*$ is the delay time of place, the default value is 0. The delay time of place $p_i$ is denoted by $c_{t_i}$.
9. $M_0 : P \rightarrow f_S$ is the initial marking of $\Sigma$.

The page of SCN presents a task and its interface is empty. Place, transition and interface describe the position of service composition, the possible operation and the input/output parameters. $\forall x \in (P \cup T)$, we denote the pre-set of $x$ as $x = \{y | y \in (P \cap T) \land (y, x) \in E\}$ and the post-set of $x$ as $x^* = \{y | y \in (P \cap T) \land (x, y) \in E\}$. The input/output arc of $t_i$ and the free variable in $A_T(t_i)$ are denoted by $FV(t_i)$.

In this paper, we abstract the resource and message as the individuality $d_i = (it, i, RW_i)$, it $\in \{w, f, d\}$ represents the described object of individuality. $w, f, d$ represent the service, fault and data packet respectively. $i$ represents the position of individuality, for example, the position of service $WS_{i,j}$ in the set of available service $WS$. When it is equal to $w$, then $RW_i$ is the attributes of service. Let $kth(d_i)$ be the $k$th element of individuality $d_i$. While the common data packet is abstracted as individuality $\varphi$, all individualities in SCN model are $\varphi$.

Definition 5: $\forall p_i \in P$, place $p_i$ has an individuality $d_j, \xi_j$ is the generated time of individuality $d_j, c_{t_i}$ is the delay time of place $p_i$, the vector: $TS(p_i) = \{\{c_{t_i} - \theta, \xi_j\}, 0\}$ is the wait time of place $p_i$.

$M^a$ is the available individuality of $M$, $M^u$ is the unavailable individuality of $M$. A tuple $S = (M, TS)$ is called the state of $\Omega$ at time $\theta$. $M$ describes the distribution of resource and $TS$ is the wait time of $M$ at $\theta$, which describes the time factor of system. Initial state $S_0 = (M_0, TS_0)$, $TS_0$ is a zero vector. The mapping $CutN : \{N_{w_p}, N_{f_p}, N_{d_p}, \ldots N_{f_p}\}$ is called the pointcut of the system, $CutN$ is the name of pointcut, $N_{w_p}$, $N_{f_p}$, $N_{d_p}$ are the corresponding joinpoint of pointcut. A triple $Asp = (CutN, AN, IN)$ is called an aspect of service composition, $CutN, AN, IN$ represent the pointcut, advice net and introduction net.

For $\forall t \in T$, if $FV(t_i) = \{x_1, x_2, \ldots, x_n\}$, the individuality set $\{d_1, d_2, \ldots, d_n\}$ meets $d_i \in \{M^a(p)| p \in \{t \cup t^*\}\}$. $d_i$ corresponds to the variable $x_i$. The instance of transition $t$ is got by replacing $d_1, d_2, \ldots, d_n$ with $x_1, x_2, \ldots, x_n$, which is called a replacement of transition $t$, denoted by $t < d_1, d_2, \ldots, d_n$. $A_T(t) < d_1, d_2, \ldots, d_n$ and $A_F(p, t) < d_1, d_2, \ldots, d_n$ are got by replacing formulas $A_T(t)$ and the predicates $A_F(p, t)$ of input arc with $d_1, d_2, \ldots, d_n$.

Let $t < d_1, d_2, \ldots, d_n$ is the replacement of $t$, if $t < d_1, d_2, \ldots, d_n$ which makes $A_T(t)$ true, then $t < d_1, d_2, \ldots, d_n$ is called the feasible replacement of $t$ under $S$. All the feasible replacements of $t$ under $S$ are denoted by $VP(S, t)$. Set $ET(S) = \{t | VP(S, t) \neq \emptyset\}$.

Definition 6: For $t_i \in ET(S)$, if $t_i$ meets the following conditions: $\lambda_i \leq \min(\lambda_j)$, where $t_j \in ET(S)$, then the firing of transition $t_i$ under state $S$ is effective. All the effective firing transitions under state $S$ are denoted by $FT(S)$.

The process that $S$ reaches a new marking $S'$ by firing a feasible replacement $t_i < d_1, d_2, \ldots, d_n$ of transition $t_i$ is denoted by $S[t_i < d_1, d_2, \ldots, d_n] \rightarrow S'$. The set $MT(S) = \{t_i | t_i \in T \land t_i \rightarrow S'\}$ is the concurrent transitions under state $S$. The set $H(S) = \{t < d_1, d_2, \ldots, d_n | t \in MT(S), \quad (t < d_1, d_2, \ldots, d_n) \in VP(S, t)\}$ is called the greatest concurrent set of $S$. 
Definition 7: The model $\Omega$ will reach a new state $S' = (M', TS')$ by effectively firing all enabled transitions in $H(S)$, $\omega$ is the delay time of $H(S)$, denoted by $S'(H(S), \omega) > S'$. $S'$ is called the reachable state of $S$, $S'$ is computed based on the following rules:

1. $\forall t_i < d_1, d_2, \ldots, d_n \in H(S), \forall p_{ij} \in t_i \cup t_i^*$:  
   \[ M'(p_{ij}) = M(p_{ij}) - A_F(p_{ij}, t_i) < d_1, d_2, \ldots, d_n > + A_F(t_i, p_{ij}) < d_1, d_2, \ldots, d_n > \]

2. $\forall d_k \in M(p_i), TS'(d_k) = e_k, TS'(d_k) = \max\{0, TS'(d_k) - \omega\}$.

If there exists the firing sequence $H_1, H_2, \ldots, H_k$ and state sequence $S_1, S_2, \ldots, S_k$, which make $S'(H_1, \omega_1) > S_1[(H_2, \omega_2) > M_2 \ldots S_k[H_k, \omega_k] > S_k$, then $S_k$ is a reachable state from $S$. All the possibly reachable states of $S$ are denoted by $R(S)$ and $S \in R(S)$.

### 3.2 Base layer model

The base layer model CN$_i$ of $C_i$ is shown in Figure 1. The specific process is (The checkpoints of component $C_i$ is $K$): The system will fire $t_{i1}$ to select an available service $WS_{i,j}$ from the service library($p_{ai}$) after getting the input parameter($M(p_{ai}) \neq \emptyset$, and makes $WS_{i,j}$ be in the running position $p_{ai}$. If there is a service which is in the running position ($M(p_{ak}) \neq \emptyset, k = 1, 2, \ldots, K$), then invoke $t_{ak}$ to do the $j^{th}$ operation for service. $t_{se}$ is used to output the final result to $p_{we}$. $p_{a1}, p_{a2}, \ldots, p_{aK}$ describe the running position of the component.

#### Figure 1 Base model of component

![Base model of component](image)

#### Figure 2 Modelling basic relationship

The basic relationships of component(sequence, loop, parallel, choice) are shown in Figures 2(a) to 2(d). For sequence relationship, we introduce $t_{ij}$ to output the result of forward component to the input of afterward component. For the loop relationship, we introduce $p_{cn}$ to control the number of loop operation, while $t_{re}$ is used to fire the repetitive operation. If all operations have operated successfully, then invokes $t_{sf}$ to fire the afterward component $C_f$. For the parallel relationship ($C_k \in Fork(C_k) \cap Fork(C_f) \cdots \cap Fork(C_f)$, $C_f \in Back(C_k) \cap Back(C_f) \cdots \cap Back(C_f)$), if $C_k$ has finished the operation, then invoke $t_{k,ij}$ to fire $C_i$ and $C_f$.

Requirement instantiation: the steps for constructing the model $\Omega$ of requirement model are:

1. Constructing the $SN$ model of all components. $P_{i1}, P_{o}$ are the input and output interface of $C_i$.

2. Introducing $t_s$, $t_e$ to describe the beginning and termination operation. $t_{di}$ and $p_o, p_{di}$ are used to control the deadline of whole application.

3. Forming the base layer model of service composition according to the model of component and their relationship.

4. Setting $M_0(p_o) = \varphi$, the priority of $t_{di}$ and all transitions in the $SN$ are set to 2, the priority of new transition is 3.
3.3 Meta layer model

**State assessment.** State assessment is mainly used to assess the state of the system. The meta layer model of state assessment is shown in Figure 3(a). \( t_{ak} \) is used to collect the execution results of component and store the results into \( p_{wpk} \). \( t_{pk} \) is used to predict the results and store them into \( p_{cpk} \) based on the input parameters \( (p_{i}, k) \). If the error is greater than the threshold, then invoke \( t_{fpk} \) to make service be in the failure position \( p_{fpk} \).

**Online monitoring.** Online monitoring is mainly used to monitor the service composition, which includes the termination and failure of component, the meta layer model of online monitoring is shown in Figures 3(b) to 3(c). For component \( C_i \), we introduce \( t_{toi} \) to send the termination info of component to \( p_{ec} \). Similarly, we introduce \( t_{ffi} \) to send the fault info of component to \( p_{fc} \). Therefore, we can monitor the operation of service composition by observing the info of \( p_{fc} \) and \( p_{ec} \).

**Fault isolation.** If component \( C_i \) fails, the fault isolation will inform the other components interacting with \( C_i \) in order to avoid the spread of failure, and the system will abort the operation of other components for recovery. If component \( C_i \) outputs the failure info and \( IC(C_i) \neq \emptyset \), then send the isolation signal [Figure 3(d)]: transition \( t_{ffi} \) is used to send the isolation signals to all components in the isolation set of \( C_i \), while the component will be in the position of waiting for recovery \( p_{wfr} \). If \( C_i \) receives an isolation signal \( (p_{fr}) \), then call \( t_{fr} \) to make the component be in \( p_{wfr} \) to wait for execution [Figure 3(e)].

**Fault recovery.** Fault recovery is used to invoke the self-healing strategy according to the different fault types, thus ensuring the normal execution of service composition.

The meta layer model of fault recovery is shown in Figures 3(f) to 3(g). If \( C_i \) fails within the deadline, the system will do the fault recovery according to the attributes of \( C_i \). If the relationship between \( C_j \) and \( C_i \) is choice, then introduce \( t_{trij} \) to invoke \( C_j \). If \( C_i \) does not have the alternative component, then introduce \( t_{tti} \) to reinvoke \( C_i \). If the fault is successfully recovered, then invoke \( t_{tfr} \) to send a signal to the isolated components. Therefore, the priority of \( t_{tti} \) is higher than \( t_{trij} \). If \( C_j \) has been reinvoked, then invoke transition \( t_{ftr} \) to reset the available services of component [Figure 3(g)].

3.4 Meta-object protocol and its modelling

**Meta-object protocol of component.** Let base layer model of \( C_i \) be \( CN_i \), the step for constructing the model of meta-object protocol of component is:

1. For the \( k \)th \((k \neq 1)\) running position \( p_{ak} \) (weaving position), we introduce a state assessment model, which is shown in Figure 3(a), where \( p_{ak}^* = t_{ak}, p_{wpc} = t_{wpk} − 1 \). If the assessment fails, then introduce \( t_{fr,k} \) to do the corresponding operation.

2. The weaving condition is: if \( C_i \in IC(C_j), C_j \in C \). For the running marking \( p_{ak} \) (weaving position) of \( C_i \), we introduce a fault isolation model of component, which is shown in Figure 3(e), where \( p_{fr}^* = t_{fr}, t_{fr} \) is used to continue the operation of component.

3. Introducing the model to reset the library of available service [Figure 3(g)], where \( t_{ftr} = p_{fw}^* \).

**Meta-object protocol of service composition.** The modelling process is: the system will add the required meta-object model to the base layer model of service
compositional according to the following steps, then merge the same place and transition:

1. Weaving rule of fault isolation: If \( IC(C_i) \neq \emptyset \), then introduce the isolation proposition \( P^i_{fr} \) (weaving position) of component \([Figure 3(d)]\). Otherwise, \( t_{fi} \) is introduced to make the component be in the fault recovery position \( p_{wfr} \).

2. Weaving rule of fault recovery: If \( C_i \) is waiting for recovery \( p_{wfr} \), then we introduce the fault recovery model \([Figure 3(f)]\).

3. For components \( C_i \), we introduce the online monitoring model \([Figure 3(b) to 3(c)]\) at the result output position \( P^i_o \) and fault output interface \( P^i_{fr} \).

4. Modelling the human intervention: If \( C_i \) outputs the timeout signal, then fire \( t_{o} \) to make service composition be in the human intervention position \( p_{so} \). Introducing \( p_{so} \) to describe the deadline of service composition, if service composition cannot finish the operation before the deadline, then fire \( t_{ao} \) to make service composition be in the human intervention position \( p_{ao} \).

The model is got by synthesising the meta-object protocol model with base layer model is called the seal-healing model \( \Omega_h \).

4 Effectiveness of self-healing strategy

4.1 Effectiveness of self-healing strategy

CTL introduces the following operators to describe the uncertainty factors, always (G), sometimes (F), next (X), until (U), all paths (A), exists a path (E). The formulas of CTL can be expressed as (normal formulas BNF):

\[ \varphi ::= \text{TRUE} | \text{FALSE} | \varphi_1 \land \varphi_2 | \varphi_1 \lor \varphi_2 | \varphi_1 \rightarrow \varphi_2 | X \varphi_1 | A \varphi_1 | E \varphi_1 | EF \varphi_1 | AG \varphi_1 | EG \varphi_1 ] \]

\( p_f \) belongs to the atomic formula set. Let \( \Omega, \Omega_h \) be the base layer model and self-healing model, which are got by using \( SCN \) to model for the requirement. \( R(\Omega), R(\Omega_h) \) are the set of reachable state of \( \Omega, \Omega_h \). Let \( S \) be a reachable state of \( \Omega, \varphi \) is a formula described by using \( CTL \). \( \Omega, S \models \varphi \) explains that \( S \) meets the formula \( \varphi \).

The self-healing model can also accurately describe the process of service composition. However, it contains the meta object protocol, thus increasing the state space of constructed model. So the base layer model is used to analyse the related properties of functional requirements in the early analysis, then further research on the self-healing model, thereby reducing the workload of analysis and computation.

Theorem 1: Let \( R(\Omega_h) \) be the corresponding reachable set. \( \forall C_i \in C, \) component \( C_i \) has \( K \) checkpoints, then \( 1 < k \leq K \), there are: \( \Omega_h, S_0 \models A(\neg p_f(C_i p_{ak+1}) \lor p_f(C_i p_{epk})) \).

Proof: (reduction to absurdity), we can suppose \( \Omega_h, S_0 \models A(\neg p_f(C_i p_{ak+1}) \lor p_f(C_i p_{epk})) \) is not established, that is, there is \( S_1 \in R(\Omega_h), \) which makes \( \Omega_h, S_1 \models p_f(C_i p_{ak+1}) \), and \( |M_1(C_i p_{ak+1})| = 1 \).

Let the state from \( S_0 \) to \( S_1 \) be the set \( R(S_0, S) \), \( \forall S_j \in R(S_0, S) \), there is \( |M_j(C_i p_{ak+1})| = 0 \).

Because \( \circ(C_i p_{ak+1} = C_i p_{ak+1}, \circ(C_i p_{epk} = C_i p_{epk}) \).

Therefore, \( \exists S_j \in R(S_0, S) \), makes \( |M_j(C_i p_{ak+1})| = 1 \) and \( C_i p_{ak+1} \in H(S_j) \).

Because \( C_i p_{ak+1} \notin \{C_i p_{ak+1}, C_i p_{ak}\} \), and the priority of \( C_i p_{ak+1} \) is higher than \( C_i p_{ak+1} \).

Therefore, we can get \( C_i p_{ak+1} \notin H(S_j) \), which is contradicted with the supposition.

Therefore, the supposition is not established, that is, \( \Omega_h, S \models A(\neg p_f(C_i p_{ak+1}) \lor p_f(C_i p_{epk})) \).

Theorem 1 explains that the component can continue operating when the results of prior checkpoint have passed the assessment. Fault isolation and fault recovery can effectively control the state of system when a failure occurs. Therefore, it is necessary to analyse whether the self-healing model can correctly isolate the fault.

Theorem 2: \( \forall S \in R(\Omega_h), \forall C_i \in C. \) If \( \Omega_h, S \models p_f(P^i_{fr}) \).

\( IC(C_i) \neq \emptyset, \forall C_j \in IC(C_i) \), there are:

\( \Omega_h, S \models G \left( EF(p_f(C_j p_{wr}) \lor p_f(C_j p_c)) \right) \)

\( \Omega_h, S \models EF(p_f(C_j p_{fr}). \)

Proof: \( \Omega_h, S \models p_f(P^i_{fr}) \), that is, the component \( C_i \) will output the fault signal under state \( S \). \( \forall C_j \in IC(C_i) \). Because \( \Omega_h, S \models p_f(P^i_{fr}) \), we can get \( M(P^i_{fr}) \neq \emptyset \).

Because \( IC(C_i) \neq \emptyset \), the system will introduce the related isolation model in the modelling process of meta object protocol. Therefore, \( \exists S_1 \in R(S) \), which makes \( M_1(C_i p_{fr}) \neq \emptyset \).

Because \( C_j \in IC(C_i) \), the component \( C_i \) may be in the running or termination position under state \( S \).

Case 1: the component is in the running position, that is \( M_1(C_j p_{wr}) \neq \emptyset \), \( M_1(C_j p_c) = \emptyset \). Because \( C_i p_{fr} = \{C_i p_{fr}, C_i p_{fr}\} \), we can get \( C_i p_{fr} \in H(S_1) \). Because \( C_j p_{fr} = C_j p_{wr} \), \( \exists S_2 \in R(S_1) \), which makes \( |M_2(C_j p_{wr})| = 1 \), that is, \( |M_2(C_j p_{wr})| \lor |M_2(C_j p_c)| = 1 \). Therefore, \( \Omega_h, S \models EF(p_f(C_j p_{wr}) \lor p_f(C_j p_c)) \) when component \( C_i \) is in the running position.

In the similar way, we can get \( \Omega_h, S \models EF(p_f(C_j p_{wr}) \lor p_f(C_j p_c)) \) when component \( C_i \) is in the termination position.

In summary, \( IC(C_i) \neq \emptyset, \forall C_i \in IC(C_i) \), there is \( \Omega_h, S \models EF(p_f(C_j p_{wr}) \lor p_f(C_j p_c)) \). That is, the sub-proposition(1) establishes.

In the similar way, the sub-proposition(2) establishes.

In summary, \( \Omega_h, S \models p_f(C_j p_{fr}) \).
Theorem 2 explains that all running components can receive an isolation signal when a component fails. So self-healing model can correctly describe the fault isolation.

Theorem 3: Let $R(\Omega_h)$ be the reachable set of model $\Omega_h$, $\forall S \in R(\Omega_h)$, $\forall C_i \in C$. If $\Omega_h, S \models p_f(P^i_1)$, there exists $\Omega_h, S \models EFpf(P^i_1)$, where $C_j \in Back(C_i)$.

Proof: Because $\Omega_h, S \models p_f(P^i_1)$, we can get $\exists S_1 \in R(S)$, which makes $M_1(p_{wfri}) \neq \emptyset$, that is, $C_i$ is in waiting for self-healing position under state $S_1$, $C_i$ has two cases under state $S$: there does not have the choice component with $C_i$ and there exists the choice component with $C_i$.

Case 1: there does not have the choice component with $C_i$, that is, $Ar(t_{fri})$ is false, therefore, $t_{fri} \neq H(S_1)$. Because $t_{fri} = t_{fri} = t_{pefri}$ and $p_{wfri} \neq \emptyset$, therefore, $t_{fri} \neq H(S_1)$. Because $t_{fri} = t_{pefri}$, we can get $\exists S_2 \in R(S_1)$, which makes $M_2(C_i, p_{fri}) = 1$. Because $C_i, p_{fri} \neq t_{pefri}$, therefore, $C_i, p_{fri} \in H(S_2)$. According to the execution process of $C_i$, we can get $\Omega_h, S_3 \models EFpf(P^i_2)$. Because $C_i \in Back(C_i)$, we can get $t_{p_{fri}} = t_{p_{fri}} \neq P^i_1$. Therefore, $t_{p_{fri}} = P^i_1$. And the priority of $t_{p_{fri}}$ is equal to 0, we can get $\Omega_h, S_3 \models EFpf(P^i_2)$. If there does not have the choice relationship with $C_i$, there is $\Omega_h, S \models EFpf(P^i_2)$.

In summary, the theorem establishes in the Case 2.

5 Examples

Based on the requirement of export service, we can construct the base layer model, meta layer and self-healing model of export service. The self-healing model of export service is shown in Figure 4, the model describes the process of export service and the execution process of meta-object. Assume that each component has 15 available services.

According to the execution process of service composition, we can get a critical set of component in export service, which is: $KC = \{C_1, C_2, C_3, C_5, C_7, C_8\}$. As the running time of parallel components may be the same, the critical set of components in the same execution process is not sole. Based on the state space of base layer model, we can get that export service can be composed, that is, there is a series of available services that can realise the function of service composition, which maps into the model $\Omega$ is: $\exists S \in R(S_0), M(p_e) = 1$.

In order to illustrate the effectiveness of proposed method. We employ the QWS dataset (Al-Masri and Mahmoud, 2003) and a self-generated dataset in the experiments. The QWS dataset contains more than 2,500 services and their quality is described by nine attributes whose values are measured upon real-life public services. First, the system will randomly select 1,560 web services from the service resource of export service. Each service at least has the basic information such as service name.
The purpose of Experiment 1 is to analyse the effectiveness of reflection mechanism. The specific steps are: Taking 840 services as service resource and dividing into 6 groups. The component in each group has 5, 10, 15, 20, 25, 30 available services, then compute the state space of base layer model and self-healing model of export service.

The experiment results of Experiment 1 are shown in Figure 5. From the figure, we can get: the state space of base layer model meets the following

1. The state space of base layer model will not increase with available services increasing, the main reason is that base layer model does not consider the failure of available service, so it is unrelated with the service resource of component.

2. The state space of self-healing model is related to the number of available services. When the available service is relatively little, the state space displays the growing trend with the number of available service increasing.

Figure 5  Experiment results of Experiment 1 (see online version for colours)

The purpose of Experiment 2 is to analyse the changing regularity of the available services nws of component, the specific steps are:

1. Taking 160 services as service resource, each component has 20 available services.

2. Let the running time of component be same, then compute the available services of component when the running time of service composition is 30, 35, 40, 45, 50 respectively.

3. Let the deadline of service composition and running time of component be stable. The checkpoint of each component is 1, 2, 3, then compute the required available services of component.

The results of Experiment 2 are shown in Figure 6. From the figure, we can get:

1. The required available services of component will display the growing trend with the deadline of service composition increasing, but it does not progressively increase. For example, component C_1 requires 8, 11, 15 available services when the deadline is 35, 40, 45. The reason is: the number of required services of component is related to the local deadline of component.

2. The required available service of component will display the growing trend with the checkpoint number increasing. The reason is that the more checkpoints, the earlier of finding the failure. Therefore, the number of required services will increase, and the state space of self-healing model will increase with the required available service increasing.

6 Related work

Some fault diagnosis and recovery approaches that analyse service composition are given in Anatoliy et al. (2008), Fan et al. (2010), Li et al. (2014) and Yang and Susan (2011). In Anatoliy et al. (2008), the authors present the results of error and fault injection into web services and analyse exception and performance as the major factors affecting fault tolerance. A framework is proposed for handling fault of service composition through analysing fault requirements in Fan et al. (2010). In Li et al. (2014), the authors introduce a case-based reasoning approach to improve the system’s fault-tolerant ability meeting the end-to-end quality-of-service QoS constraints. A recovery algorithm for service execution failure in the context of concurrent process execution are proposed in Yang and Susan (2011). Although the research outcomes in the above researches are promising. Several aspects differentiate our approach from the above approaches. First, the developed self-healing service composition is easier to use because of the high abstraction level offered by using reflection mechanism. Second, we propose the self-healing strategy of component and service composition. Finally, we adopt CTL and related theories of Petri net to analyse the effectiveness and correctness of proposed method.

There are many researches on using AOP to analyse the software development, such as Geri et al. (2009) and Nafees (2009), but few papers focus on using AOP for self-healing in the context of service composition. We are aware only of the work presented in Mohamed-Hedi et al. (2011), who designed and implemented a self-healing approach to achieve web services reliability. An interesting related approach synthesises different monitoring techniques, based on aspects, to provide recovery capabilities for compositions (Karastoyanova and Leymann, 2009). The approaches do not consider how to construct the invoked web services for component. In fact, service capabilities and availability may change frequently, and the composition schema may emerge dynamically based on emerging customer requirements. In addition, we consider the deadline of service composition, and decompose it to each component, thus avoiding the endless loop in self healing process.
The work closest to ours is presented in Guan et al. (2008). In this work, the authors present a Petri net-based approach for supporting aspect-oriented modelling, but they did not consider the self-healing function of service composition. The authors in Mohsen et al. (2012) propose a more pro-active self-healing mechanism that uses a multi-layer perceptron ANN and a health score mechanism to learn about the occurrences of failures or quality of service degradation. However, they did not involve the recovery mechanism of service composition, and the method is used to implement self-healing function at the code level. Chattrakul and Jun (2013) introduced an agent-based service composition framework to allocate to the tourist an optimal composite service. Later Wang et al. (2012) proposed a self-adaptive QoS-aware reservation management approach, which calculates the reservation acceptance gain to make the final decision. Yan et al. (2013) introduced an self-adaptive integrated algorithm, which focus on the interaction scheme between the dimensionality reduction and the cluster evolution. However, they do not include how to interleave monitoring, diagnosis, and adaptation as part of the self-healing process without interrupting the execution of the unaffected web services.

7 Conclusions

In this paper, we have modelled and analysed self-healing systems using Petri nets and reflection mechanism. The special features of the proposed model include:

1. a service composition net is defined as a unified formalism to describe different components of service composition. The deadline of service composition is taken into account in the self-healing process

2. flexible self-healing strategies are proposed for component and service composition, and the related operations of self-healing are abstracted as meta-object by using reflection mechanism. Petri nets are used to construct the base layer model, meta layer model, meta-object protocol and other components

3. CTL is used to convert the self-healing strategy into the corresponding CTL formulas, and the basic properties of model are analysed.

The enforcement algorithm is proposed, and the operational semantics and related theories of Petri nets help prove the effectiveness and correctness of the algorithm. Finally, we conducted several experiments. Experiment results show that the proposed method can effectively improve the self-healing ability of service composition.

Acknowledgements

The work is partially supported by the NSF of China under grants No. 61173048 and 61300041. Research Fund for the Doctoral Programme of Higher Education of China under Grants No. 2013074110015. Innovation Programme of Shanghai Municipal Education Commission under Grant No. 12YZ166. The Fundamental Research Funds for the Central Universities under Grant No.WH1314038.

References


