A Tree-Based Reliability Analysis for Fault-Tolerant Web Services Composition

Yanjun Shu[1(✉)], Decheng Zuo[1], Hongwei Liu[1], Quan Z. Sheng[2], Wei Emma Zhang[2], and Jian Yang[2]

[1] School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
{yjshu,zuodc,liuhw}@hit.edu.cn
[2] Department of Computing, Macquarie University, Sydney, Australia
{michael.sheng,w.zhang,jian.yang}@mq.edu.au

Abstract. Reliability is critical for choosing, ranking and composing Web services. However, some common situations, such as fault-tolerant strategies and the dynamic operational profile, are not considered in existing reliability analysis. To solve these problems, a tree-based composition structure model is proposed, which is called the Fault-tolerant Composite Web Services Tree (FCWS-T). We separate the nodes in FCWS-T into two types, namely the *control nodes* and the *service nodes*, leading to the representation of various composition structures can be explicitly performed. Then, a reliability simulation method is proposed based on FCWS-T and it can effectively analyze the reliability of a complex Web service. Experiments on a financial management service show the effectiveness of our approach for fault-tolerant Web service compositions.

Keywords: Reliability · Services composition · Fault-tolerant · Simulation

1 Introduction

Nowadays, Service-Oriented Computing (SOC) has emerged as a new way to develop extensible computing systems that evolve from the component-based software engineering. In SOC, the service is a black box to users and it is either an atomic Web service or a complex Web service that is constituted by several smaller, loosely coupled, reusable Web services via the Business Process Execution Language (BPEL) [5]. Reliability is a key issue of Quality of Service (QoS) for choosing and compositing Web services [9], especially for the mission-critical domains such as military or finance. In these domains, systems are complex and built by many component services with different reliabilities, leading to the analysis a very challenging yet crucial task. To perform the reliability analysis of composite Web services, there are two main issues to be resolved:

Modeling the composition structure. An appropriate representation of the composition structure is the foundation for reliability analysis. Most existing reliability analysis methods assume that the composite Web service is well-structured by some methodologies such as Service graph [7] and Semi Markov Process (SMP) [9]. However, clear explanation on how the structure model is built from the service composition is either missing or insufficient. In practice, the composition structure is varied in the integration stage and some composite Web services may be black boxes to users. Thus the transition from a composite Web service to the composition structure model requires explicit discussion. As the BPEL process describes the service composition, the problem of modeling the composition structure can be turned into the transition from BPEL to a composition structure model [4]. Moreover, Web services operate in an unstable Internet. Fault tolerance is an effective way to achieve high reliability. Although some existing reliability analysis methods consider the fault-tolerant mechanism in reliability calculation, they do not represent the fault-tolerant strategies in their composition structure models [4,9].

Calculating the composite reliability. *Composite reliability* is the integration of *component reliabilities* with the transition probabilities between every component service. The transition probability can be obtained by statistical analysis of service invocations or empirical study of similar service compositions. All transition probabilities in a composition constitute the service operation profile which is a description of the generated pattern of external service requests expressed in a probabilistic form. Many *composite reliability* calculation methods use various mathematical equations to integrate the *component reliabilities* with the high level composition structure model [1,7,9]. These methods can obtain the *composite reliability* directly and they are applied widely in QoS-based service compositions. However, there are many restrictions on mathematical equations, such as the calculation equations may be very cumbersome and the sensitiveness of components cannot be obtained easily. Moreover, the *composite reliability* is dependent on the operation profile [2]. For a composite Web service, the operation profile may be varied in different time intervals according to users' requests. Although the dynamic operation profile is very important in reliability analysis, it is considered by few composite reliability calculation methods.

Based on above discussions, a tree-based reliability analysis approach is proposed in this paper. We represent the composition structure in a Fault-tolerant Composite Web Services Tree (FCWS-T). There are two types of nodes in FCWS-T: the *control node* and the *service node*. The service node is a leaf of FCWS-T which represents a component service. The control node is the internal node which is used to represent the composition activity of children. By separating the node types of FCWS-T, various structures of the composite Web service can be represented explicitly. Moreover, the FCWS-T can be transformed from the BPEL process or the composition designer's description directly. Considering the limitations of mathematical equations, the discrete-event simulation method [3] is used here for its flexibility in describing the *component reliability* functions. By integrating multiple operation profiles in simulation, the varying operation profile can also be considered in the *composite reliability* analysis.

The remaining paper is organized as follows: Sect. 2 presents the FCWS-T model and the methodology to transform the BPEL to a FCWS-T; Sect. 3 describes the reliability analysis simulation algorithms; Sect. 4 reports the experiments on a finance management service; Sect. 5 provides some conclusions.

2 The FCWS-T Model

2.1 The Definition of a FCWS-T Model

The FCWS-T is defined as a tree in this work. There are two main types of elements in the composition structure: component services and composition activities. Correspondingly, we define two types of nodes, namely *ServiceNodes* and *ControlNodes*, to represent them respectively. The *ControlNodes* represent four basic composition activities which include *Sequence*, *If*, *While/Repeat* and *Flow* [5]. The *ServiceNode* describes a component service's reliability and execution time. In reality, the round trip of invoking a component service is more vulnerable than the service execution. In FCWS-T, the link reliability and link time of a component service are considered in the *ServiceNode*. Moreover, only several key component services in a whole composite Web service will be fault-tolerant due to the fault-tolerant strategies application costs significantly in time or resources. Thus in FCWS-T, fault-tolerant strategies are only defined for the *ServiceNodes*. According to the classification in [8], there are three main fault-tolerant strategies: *Retry*, *Active replication*, *Passive replication*.

According to the iteration feature of a tree, the following is the definition of FCWS-T model. Every tree node is: $TreeNode = \langle type, parent, childList, weight\rangle$.

(1) *type*: The *ServiceNode* and *ControlNode*. The *ServiceNode* is $\{ServiceReli(), ServiceTime(), LinkReli(), LinkTime(), FT\}$, and $FT \in \{None, Retry, Passive, Active\}$. The *ControlNode* is $\{Sequence, If, Flow, While/Repeat\}$.
(2) *parent*: FCWS-T *TreeNode*, the father of the tree node.
(3) *childList*: $\{child_1, child_2, \cdots, child_n\text{: } FCWS-T\, TreeNode\}$.
(4) *weight*: The execution probability p_i relative to the parent node. In the *If* activity, $\{p_i\}$ is the branch execution probability. In the *While/Repeat* activity, p_i represents the probability of executing i times. In both of these two activities, the sum of all branch execution probabilities is 1. In the *Sequence* or *Flow* activity, all children execute in sequence or in parallel and p_i is 1 for the children.

2.2 The Transition from BPEL to FCWS-T

The BPEL process of a composite Web service elucidates the structure activities (i.e., a series of basic composition activities) by nesting and iterations [5]. Here, we build the FCWS-T model directly by parsing BPEL process in two steps.

(1) Extracting the WS-token String from BPEL

We define the WS-token string to represent the lexical analysis results of BPEL. A WS-token string is a set of tuples. Each tuple represents a service sub-composition and it is consisted by four elements: the *left bracket* "(", the *basic composition activity*, the *Web service number*, and the *right bracket* ")". The *left bracket* "(" and the *right bracket* ")" denote the start and the end of a sub-composition activity. The *basic composition activity* can be *Sequence*, *Flow*, *While/Repeat*, *If* and they are denoted as S, F, W, I. The *Web service numbers* are the identifiers of component services invoked in the sub-composition of a tuple.

The extraction process includes three parts. First, the BPEL source file is split into strings by lexical analysis. Then, the strings are read in sequence and the corresponding element of a tuple is generated. For example, in a sequence sub-composition, there are two component services which are Service 1 and Service 2. The tuple of this sub-composition is denoted as (S12). Finally, by parsing all BPEL strings, a WS-token string is created by constituting the tuples nested.

(2) Mapping the WS-token String to FCWS-T

As an intermediate representation, the WS-token string can be used for transforming BPEL to FCWS-T. Every tuple of the WS-token string represents a Web services sub-composition. Algorithm 1 shows the mapping algorithm from the WS-token string to FCWS-T. The WS-token string is scanned from left to right. A sub-composition starts with the *left bracket* "(" and ends with the *right bracket* ")". The new tree nodes of the *ControlNode* and *ServiceNode* will be created according to the *basic composition activity* and *Web services number* of a tuple. When a sub-composition activity finishes, the corresponding subtree is generated and inserted to FCWS-T as a component service.

Algorithm 1. MapFCWS-T

```
Input: a WS-token string;
Output: the FCWS-T;
1.  current=0;
2.  while (current <WS-token.length)
3.   { current++;
4.     if(WS-token[current]== Composition )
5.       S1.push(WS-token[current]); //S1 is a composition activity stack.
6.     elsif(WS-token[current]== "(" )
7.       S2.push(WS-token[current]); //S2 is a service stack.
8.     elsif (WS-token[current]== Number )
9.       S2.push(WS-token[current]);
10.    elsif (WS-token[current]== ")") // A sub-composition activity is ended.
11.      {Con_node=S1.pop(); New_tree=Create_tree(Con_node); //The ControlNode is generated.
12.       Ser_node=S2.pop();
13.       while(Ser_node != "(")
14.          {Insert_Node(New_tree, Ser_node); Ser_node=S2.pop(); } // ServiceNodes are inserted.
15.       S2.push(New_tree); } // The subtree is pushed in the service stack as a component service.
16.    end if; }
```

3 The Reliability Analysis Simulation Methodology

To calculate the reliability of a service composition, we need a mechanism which can integrate the composition structure model and component reliabilities. The simulation method is an effective way to address these two issues. Moreover, it can explore the "what-if" questions and get more reliability details at the design stage [3]. Here, the discrete-event simulation is adopted to study the failure behavior of each component service in the composition. Then, a simulation algorithm for the whole composite Web service is proposed based on the FCWS-T.

3.1 The Discrete-Event Simulation of Component Reliability

The discrete-event simulation technique [3] has been used to study the failure behavior of Web services which are described by a non-homogeneous continuous time Markov chain (NHCTMC) process. The failures of a Web service are treated as the discrete-events in simulation. The main idea of this technique is to compare a random number x with the probability of a failure occurred (i.e., a event happens) in the infinitesimal interval $(t,\ t + dt)$. The failure probability is given by $lambda() \times dt$ and $lambda()$ is the failure rate function, which can be provided by service developers or the evaluating third party. If $x > lambda() \times dt$, it means a failure happened in $(t,\ t + dt)$ and returns 1, otherwise the service executes successfully and returns 0. The Web service reliability can be obtained by the number of failures is divided by the entire simulation times in the period $(0,t)$.

It is costly and not feasible to explore every fault tolerant strategy via testing. The simulation technique can help developers in determining how fault-tolerant Web services will perform when they are employed. In our previous work [6], we have applied the discrete-event simulation method to investigate the reliability problem of fault-tolerant Web services. The reliability simulation algorithms of retry, active replication and passive replication strategies are proposed. Due to space constraints, the details of these simulation algorithms are not discussed.

3.2 The Simulation Algorithm of the Composite Reliability

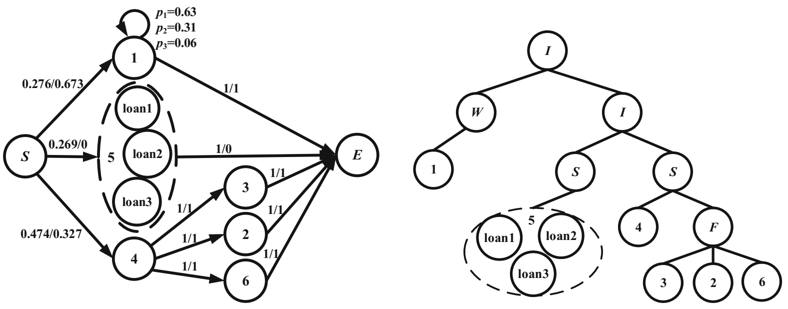
As the composition structure and component services are distinguished by *ControlNodes* and *ServiceNodes*, the composite reliability simulation just needs to travel FCWS-T according to the type of tree nodes. Algorithm 2 shows the simulation process of composite services. The basic idea of our algorithm is to travel all sub-trees in a preorder. Each sub-tree from the root node is iteratively simulated according to the composition structure of their father node. When the tree node is a *ServiceNode*, the component reliability simulation is executed. The link reliability and service reliability are simulated sequentially for a *ServiceNode*. If a service or link is failed, the simulation stops. The failure times and the execution time are recorded. Otherwise, the simulation will traverse all nodes in FCWS-T and return the execution time.

Algorithm 2. SimulateReli

```
Input: The FCWS-T, n;
Output: linkfails[], servicefails[], exetimes[], globatime;
1.  SimCounting=0; globatime=0;
2.  while (SimCounting <n)
3.  {SimuCounting++;
4.    TreeNode=FCWS-T.root; localtime=globaltime;
5.    while(TreeNode !=NULL || failureTag==FALSE)
6.    { if (Treenode.type is ControlNode)
7.        switch(TreeNode)
8.          case "S": foreach Subtree_i do SimulateReli (Subtree_i) in sequence; break;
9.          case "I": foreach Subtree_i do SimulateReli(Subtree_i) in branch; break;
10.         case "W": foreach Subtree_i do SimulateReli(Subtree_i) in loop; break;
11.         case "F": foreach Subtreei do SimulateReli(Subtreei) in parallel; break;
12.     elsif (TreeNode.type is ServiceNode)
13.           failureTag=Link_Service_Sim(TreeNode,localtime);
14.           Update linkfails[], servicefails[], exetimes[], localtime;
15.           return failureTag;
16.     end if; }
17. globaltime+=localtime; }
```

Table 1. The reliability of component Web services

No.	Service Name	Exeution Time avg (ms)	Reliability
1	Deposit and withdraw	104.4	0.782
2	Intermediate approval	103.17	0.863
3	Primary approval	95.02	0.983
4	Risk assessment	91.47	0.792
5	Loanversion1	88.28	0.804
5	Loanversion2	97.56	0.793
5	Loanversion3	90.46	0.788
6	Advanced approval	127.3	0.887



(a) Operation profiles (b) FCWS-T

Fig. 1. The operation profiles and the FCWS-T model of the financial management composite service

4 Experimental Studies

4.1 The Experiment Setup

A financial management composite service is used to demonstrate the effectiveness of our reliability analysis approach. This composite service provides the deposit and withdrawal service, the investment service and the loan service. The investment service is composed by four component services which are the risk assessment service, the primary approval service, the intermediate approval service and the advanced approval service. Moreover, the passive fault-tolerant strategy is applied for the loan service to ensure its reliability. There are three loan services which are named `loanversion1`, `loanversion2`, `loanversion3`. The reliability of each component service is shown in Table 1. As the loan service is not available in the non-working hours, the working hours operation profile is quite different from the non-working hours. 14,925 test cases are executed during the period of one month. The numbers of test cases in the working hours and non-working hours are 10,031 and 4,894. These two groups of test cases constitute the working hours and non-working hours operation profiles which are shown in Fig. 1(a).

4.2 The Simulation Reliability Analysis Results

This section reports the results of the simulation approach and it is twofold. First, we exhibit the usability of simulation results with multiple operation profiles. Second, we demonstrate how the simulation approach determines the reliability bottleneck and explore the effectiveness of different fault-tolerant strategies.

(1) The Reliability Simulation Results

The FCWS-T model is generated by transforming the BPEL of the financial management service. First, the WS-token string is extracted from the BPEL and it is $(I(W1)(I(S1)(S4(F326))))$. Then, the FCWS-T is generated. Figure 1(b) shows the FCWS-T of the financial management service. Based on Table 1 and Fig. 1(a), the parameters can be specified for the *ServiceNodes* and *ControlNodes* respectively. In our examples, the $LinkTime()$ of services is a random value which ranges from $0ms$ to $200ms$ and the $LinkReli()$ of services is set as 0.99 since the financial management service is operating in a small local area network.

As the test cases of working hours and non-working hours are 10,031 and 4,894, the proportion of two operation profiles execution can be assumed as 2:1. We define that every 1,000 simulations of the working hours will follow 500 simulations of the non-working hours. The two operation profiles are alternatively simulated. With 100,000 simulation times, the average reliability and execution time are 0.7383 and 254.84 *ms*. The simulation reliability results of working hours and non-working hours are 0.7541 and 0.6762. As the executions of the working hours profile are twice of the executions of the non-working hours profile, the whole time result is more close to the working hours. Moreover, the reliability of the non-working hours is much lower than the reliability of the working hours. The simulation results suggest that developers need to pay more attention on the reliability of the financial management service in non-working hours.

(2) The Fault-tolerant Strategy of Web Services

Finding the most reliability sensitive component service is essential in applying fault-tolerant strategies. The sensitiveness of every component service can be investigated by changing component reliabilities. When every component reliability is increased by 10% in each composite reliability simulation, Service 1 is found to be the most sensitive component service which has the greatest improvement of the composite reliability. Thus it is an effective way to improve the whole composition reliability by applying fault-tolerant strategies on Service 1.

With the simulation approach, we can further explore the effectiveness of fault-tolerant strategies in improving the reliability of Service 1 and the whole composition. For *Retry* strategy, Service 1 will repeat three times until it succeeds. For *Passive* strategy, three replicas of Service 1 will be executed in order if the prior one is failed. For *Active* strategy, three replicas of Service 1 are executed in parallel. The execution result is the first return of three versions. Each replica is configured with different reliability and execution time. Table 2 shows the simulation results of Service 1 and the whole composition with different fault-tolerant strategies. It can be seen that the reliability of Service 1 is significantly improved by applying fault-tolerant strategies. However, the resources and execution time are also increased. The whole composite reliability can be improved by 14.7%, 16.3% and 16.1%, comparing with no fault-tolerant strategy of Service 1. The composition designer can choose a suitable strategy to improve the reliability of the whole composite Web service based on the simulation results.

Table 2. The Reliability Results of Service 1 with Different Fault-Tolerant Strategies

Attributes		The fault tolerant strategy of Service 1			
		Non_FT	***Retry***	***Passive***	***Active***
Service 1	Resources	1	1	3	3
	Execution Time avg (ms)	104.4	235.09	233.88	206.23
	Reliability	0.782	0.9906	0.9927	0.9924
Whole composition	Execution Time avg (ms)	254.84	290.66	287.98	277.43
	Reliability	0.7383	0.8464	0.8586	0.8578
	Reliability improved	0%	14.7%	16.3%	16.1%

5 Conclusion

This paper proposes a tree-based reliability analysis approach for fault-tolerant Web services composition. The composition structure is represented by the

FCWS-T model which is a tree. Based on the FCWS-T model and the discrete-event simulation method, the composition structure, the component reliabilities and fault-tolerant strategies can be integrated in the composite reliability analysis. Developers can not only obtain the reliability of the whole composite Web service with multiple operation profiles, but also the sensitiveness of each component Web service and the effectiveness of different fault-tolerant strategies.

Acknowledgement. This work is partially supported by China NSF (No. 6120209 1), the Fundamental Research Funds for Central Universities (No. NSRIF. 2016050) and the State Scholarship Fund of China Scholarship Council (No. 201606125073).

References

1. Ding, Z., Jiang, M., Kandel, A.: Port-based reliability computing for service composition. IEEE Trans. Serv. Comput. **5**(3), 422–436 (2012)
2. Grassi, V., Patella, S.: Reliability prediction for service-oriented computing environments. IEEE Internet Comput. **10**(3), 43–49 (2006)
3. Lin, C.: Analyzing the effect of imperfect debugging on software fault detection and correction processes via a simulation framework. Math. Comput. Model. **54**(11), 3046–3064 (2011)
4. Mukherjee, D., Jalote, P., Gowri Nanda, M.: Determining QoS of WS-BPEL compositions. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 378–393. Springer, Heidelberg (2008). doi:10.1007/978-3-540-89652-4_29
5. OASIS: Web Services Business Process Execution Language (WS-BPEL) v2.0. (2007), http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
6. Shu, Y., Wu, Z., Liu, H., Gao, Y.: A simulation-based reliability analysis approach of the fault-tolerant web services. In: Proceedings of ISMS 2016 (2016)
7. Zheng, H., Yang, J., Zhao, W.: Probabilistic QoS aggregations for service compositions. ACM Trans. Web **10**(2), 1–36 (2016)
8. Zheng, Z., Lyu, M.: A distributed replication strategy evaluation and selection framework for fault tolerant web services. In: Proceedings of ICWS 2008 (2008)
9. Zheng, Z., Trivedi, K., Qiu, K., Xia, R.: Semi-markov models of composite web services for their performance, reliability and bottlenecks. IEEE Trans. Serv. Comput. **6**(1), 1–14 (2015)