A Colored Petri Nets Based Model and Verification for Services Composition

Hasan A.H Naji 1, Qingji Xue 2*, Lingxiao Zhang 3, Ke Zheng 4
School of Computer and Information Engineering
Nanyang Institute of Technology, Chang Jiang Road No 80, 473004, Nanyang, China
1hasanye1985@gmail.com
2*Correspondence Email: xue_qj@sian.com, Tel: 0377-62076339
3nitzlx@163.com

Abstract: Service composition process generates composite services in order to fulfill service consumer’s requirements that cannot be satisfied by a single service. Literature review addressed services composition but ignored verifying the existence of several serious issues, which may affect Service composition and may lead to failure of the given composite services in the execution time, including consistency of the functionality and QoS Criteria. This paper adopts Colored Petri Nets based model for Services Composition and proposes a QoS aware algorithm for verifying the consistency of composite services. A case study is provided for demonstrating the applicability of the proposed model and algorithm using concepts and values of QoS Criteria of composite services.

1. Introduction
Services Composition attracts more and more attention as it plays important role in service oriented business especially in cloud services. There is a need to compose services together in order to fulfill requirements that cannot be satisfied by a single service. Verification of generated composite Services can assist on avoid serious issues may occur in execution time. These issues include Deadlock, Liveness, Reachability, etc., which may be ignored in design time. Many research and studies addressed services composition, and here is an overview of various verification approaches and tools adopted for verifying services composition.

Wibowo et al [1] proposed a fuzzy multi-criteria group decision making method for evaluating the performance of Cloud services in an uncertain environment. In the proposed method, intuitionistic fuzzy numbers are used to better model the subjectivity and imprecision in the performance evaluation process.

Deng et al [2] proposed a reliability calculation method for Web service compositions, which uses Fuzzy Reasoning Colored Petri Net (FRCPN) to verify the Web service compositions. The method presented a definition of semantic threshold similarity for Web services and a formal definition of FRCPN. In addition, the method analyzed five kinds of production rules in FRCPN, and applied the method to an SCP prototype.

Hou Jinkui and Wang Lei [3] presented formal model for web service composition and verification. The model was defined based on a framework of category theory. Process Algebra was introduced into the framework to describe the external behavior of service component, establishing a formal semantic model for the architecture of Web service system.
In [4], authors proposed an automated approach for verifying the functional requirements of a service consumer using semantics of services composition. In addition, a model checker was developed to facilitate the process of verification. The approach took advantage of labeled transition systems (LTSs) by using the semantics of BPEL programs for functional verification. Zahoor E, Perin O and Godart C [5] introduced an approach to verify the services composition processes using a method called satisfaction-solving method. The satisfaction-solving approach uses the nature of declarative processes that are defined by specifying the constraints that determine the boundary of the solution of the composition process.

SPIN [6] is a tool used for verification of distributed systems. The inputs to SPIN is a program written using Promela, a meta-language for processes and their properties are specified using linear temporal logic. The tool examines whether the program can satisfy the stated requirements.

The authors in [7] introduced a services composition model and its verification algorithm based on interface automata. The model provides semantic description of services by the extension of the interface automata. In addition, transformation rules between BPEL description and the semantic service interface automata model is defined.

Process Algebra [8] is a set complex mathematical based languages with well-defined semantics that facilitate describing and verifying the properties of concurrent communicating systems. They are modeled by means for describing labeled transition applications. LTSA (Labeled Transition System Analyzer) is a popular tool used for Finite State Process FSP model that traverses the state-space defined by a model and generates reports about first safety or progress errors that may encounter. FSP [9] used in LTSA and efficiently adopted only for the design time validation.

In [10], authors proposed a CPN based model and verification for services composition. This model illustrates types of control construct and introduces software agents works together with OWL-S specification. However, they did not mention QoS consistency, which plays an important role in composition verification.

Literature review shows there is a need to check the consistency of QoS criteria of the generated composite services in order to avoid problems that may arise in execution time and affects related users and applications.

The main contribution of this paper is presenting a Colored Petri Nets based model for services composition. The component services and their composition construction types in composite services are transferred from OWL-S into Colored Petri Nets (CPN) model using a tool called CPN Tools. In addition, an algorithm is defined to verify the QoS Criteria consistency of services composition.

The rest of the paper is organized as follows: section 2 presents a Literature Review related to the study. Section 3 introduces the proposed model and algorithm in detail. Section 4 presents a case study of implementation of the proposed model. In section 5, we summarize the content and the findings of this study.

2. Literature Review

2.1 Verification of Services Composition

Verification of Services Composition plays a vital role in order to avoid serious issues may occur in execution time including Deadlock, Liveness, Reachability, etc. The following is a list of some works of verification involved PI Calculus, Petri Nets and Automatic Machine.

(1) PI Calculus: Yuan Zhang et al. [11] studied the issue of behavioral modeling in a service composition. In their work, they adopted timed PI calculus to model service behaviors and interactions in a formal way. To deal with the structural composition, they extended the syntax and semantic of PI Calculus. In [12], an Extended Process Calculus for Cloud Services Composition is presented. The syntax and operational semantics were investigated.

(2) Petri Nets: Yue Ni, Yu Shan Fan [13] defined a formal model for verifying semantic services compositions based on Petri Nets model. Moreover, three algorithms are defined for verifying
reachability and semantic consistency. In [14], a Logical Petri Net-based Model is defined for Service Cluster Composition and Soundness Verification.

(3) Automatic Machine: Jamal Bentahar et al. [15] defined a model for checking Services Composition based on operational behaviors. The model is checked against properties defined in the control behavior. The work examines properties such as soundness and completeness of the design model according to the operational and control behaviors. An automatic testing tool called WSOTF [16] is developed for Services Composition Verification. The testing tool includes an online testing algorithm, which integrates ideas of test execution, and test debug is used to produce and execute the test cases.

2.2 Petri Nets and Colored Petri Nets

A Petri Net [17] is a bipartite directed graph, which contains two kinds of nodes called places and transitions. In addition, there is a set of arcs, which links the places with transitions or transitions with places, and there are tokens that are located in places in order to show the current state of the net. A composite service is presented by an ordered set of the component services that are mapped into a Petri Net. The mapping process is performed by determining services’ operations with transitions along with the state of service operations and places. The places, transitions and related components in petri nets are defined as the following mathematical model.

**Definition 1**: A Petri Net can be defined as the tuple: PN= (P, T, A), where: (1) P is a non-empty set of places. (2) T is a non-empty set of transitions. (3) A is a set of directed arcs which connect the places and transitions, and A ⊆ (T × P) ∪ (P × T).

As Petri Nets model has powerful features, including asynchronous, concurrent, process interoperability, etc., therefore, Petri Nets model has been widely adopted in software engineering, industrial control and many other fields. However, Petri Net model has a limitation in data concept; all data should be transferred into places and transition, and then system model becomes huge. Then Jensen [17] added a Color concept and extended the traditional Petri Nets into Colored Petri Nets.

3. Colored Petri Nets based Model and Verification for Services Composition

3.1 Modeling Services Composition in Colored Petri Nets

In this section, a CPNs model based on OWL-S [18] definition is adopted to model the component services included in a composite service. Firstly, we introduce an Algebraic Model for composite services. Secondly, the structural description of services composition in Colored Petri Nets model is presented. Finally, the Composition Constructions on Colored Petri Nets model are introduced.

1) Algebraic Model for Composite Services

The process of composing two or more services together is to generate a new value-added service, i.e., composite service, and this is performed according to a specific algebraic model between the component services. The typical operators, including as Sequence, Choice, Parallel, and Iteration are the basis for describing the control flow among component services in a composite service. Formally, the Algebraic Model for a composite service is represented as the following:

\[ S_{net} = S_1 \Delta S_2 | S_3 \Phi S_4 | S_5 \Psi S_6 | S_7 | ... | S_n \]  

where:

(1) \( S_i \) (where \( 1 \leq i \leq n \)) represents an atomic service.

(2) \( \Delta \) denotes a Sequence operator. A Sequence operator of two services \( S_1 \) and \( S_2 \) indicates producing a composite service by the sequential execution of the two services.

(3) \( \Phi \) indicates a Choice operator. A Choice operator indicates that according to the choice of a certain condition, a composite service can be constructed by choosing a service \( S_3 \) or service \( S_4 \) for execution.

(4) \( \Psi \) denotes a Parallel operator. A Parallel operator indicates producing composite service by parallel execution of more than one service.

(5) \( \delta \) indicates an Iterative operator. An Iterative operator represents producing composite service by repeating the execution of a service \( S_i \) until reaching a certain condition.
2) Structural Description of Service Composition in Colored Petri Nets Model

As mentioned above, Colored Petri Net are the extended form of traditional Petri Nets. Colored Petri Nets model adds colors and types concepts in order to describe complex data. In addition, Colored Petri Nets model integrates the advantages of traditional Petri Nets model and the advantages of advanced programming languages in order to describe, build, and analyze distributed systems. Therefore, this model becomes suitable for modeling service composition and applications in collaboration environments.

In this study, we use the formalization of Colored Petri Nets to describe composite services, and its corresponding support tool: CPN Tools is used for illustrating the composite services graphically.

Definition 2: A Services Net \([17]\) in CPN model is represented as tuple \(SN = \{PN, In, Out\}\), where:

1) CPN indicates a colored petri net.
2) In denotes input places set.
3) Out denotes output places set.

Definition 3: a composite service is represented by the tuple \(CS = \{S_{set}, , SN\}\), where:

1) \(S_{set}\) is the set of services that forms a composite service \(CS\). If \(S_{set}\) has only one member: \(S_{set} = \{s\}\), then \(CS\) is an atomic service, otherwise \(CS\) is a composite service.
2) denotes the composition construction types, and its values are obtained from ControlConstrust set \{Sequence, AND-Split, If-Then-Else, Iterate, Repeat-While, Repeat-Until\}, where Sequence is used in sequence composition construction, AND-Split is used in Parallel composition construction, If-Then-Else is used in Choice composition construction, and Iterate, Repeat-While, Repeat-Until are used in Iterated composition construction.
3) \(SN\) is the model of a composite service in Colored Petri Nets model.

In OWL-S, an atomic service usually corresponds to a process. In services composition, an atomic service and composite service are modeled as a composite process. Figure 1 illustrates an atomic process in a Petri Net model. It has only one transition called Process1. To make it simpler, we suppose it has only an input \(In\) and an output \(Out\).

In a Colored Petri Nets model, each place has a color set which specifies the values of the tokens. In Figure 1, the places \(In\), \(Out\) are included in a set of places \(P\) whereas their colored sets \(NAME\) and \(TEL\), respectively are included in the set of colored sets \(ColSet\). Transaction Process1 presents the service which has an input and an output \((In, Out)\).

3) Composition Constructions in Colored Petri Net Model

A composite service is modeled as a set of atomic processes with composition constructions that connect these processes. Based on the analysis of the typical operators and the set ConstructControl of the control flow mentioned above, there are four types of Composition Constructions. In the following, several Composition Constructions of services composition, which illustrate mapping and transformation composite services from OWL-S into Colored Petri Nets, are introduced and explained using CPN tools.

Figure 1 Atomic Process in a Petri Net
(1) Sequential Construction: It defines a set of processes that execute in sequence. Process1 and Process2 (white squared shapes) are executed in a sequential order, whereas black ones indicate control transitions, which used to execute combinational operations as shown in Figure 2-a.

(2) Choice Construction: It indicates choosing a process from given processes to be carry out. In Figure 2 Process1 and Process2 can be selected, once one of them is selected, the other must be abandoned as in Figure 2-b.

(3) Parallel Construction: Parallel Construction, indicates that two or more service operations are executing at the same time. As shown in Figure 3-a, Process1 and Process2 are executed concurrently and the other processes are executed only after the execution of the processes (Process1 and Process2) is completed.

(4) Iteration Construction: In iterated construction, OUT1 is checked, if the condition FLAG is satisfied, a transition process1 is executed until the condition FLAG is not satisfied (NOT FLAG) and then the current loop is terminated, and the next process is executed as in Figure 3-b.
3.2 Verifying of Services Composition in Colored Petri Nets

In this section, we describe Verifying QoS Criteria of services composition. The QoS Criteria of composite services are verified by finding the optimal composite service, which is compared with the optimal composite service that is found in advance. Algorithm 1 illustrates the Verifying process of QoS Criteria of services composition.

Algorithm 1: Verifying QoS Criteria

Input: $\text{CS}_{\text{set}}$ is a set of composite services; $\text{CS}_{\text{optimal}}$ is a composite service in $\text{CS}_{\text{set}}$ that is known as the optimal composite service in advance; $W$ is a set of QoS weights; $\text{Const}_{\text{global}}$ is a set of global QoS constraints.

Output: True or false, finding the optimal composite service in $\text{CS}_{\text{set}}$ and determine if it is the same composite service $\text{CS}_{\text{optimal}}$ as supposed to be.

VerifyingQoScriteria($\text{CS}_{\text{set}}, \text{CS}_{\text{optimal}}, W, \text{Const}_{\text{global}}$)

1. $\text{CS}_{\text{maxutility value}} \leftarrow \text{null}$; // the initial value of the composite service which has the maximum Utility Value is null
2. $\text{global verified CS} \leftarrow \text{null}$; // a set of composite services which satisfy global constraints
3. for ($u=1; u<o; u++$) // for all composite services in $\text{CS}_{\text{set}}$, $o$ is the number of composite services in $\text{CS}_{\text{set}}$
   4. find the composition construction type of services in of $\text{CS}_u$
   5. for ($k=1; k<5; k++$) // 5 QoS Criteria
      6. compute the aggregated value of $\text{CS}_u.c_k$
      7. if $\text{CS}_u.c_k$ doesn't violet the global constraint $c_k$
      8. continue;
Algorithm 1 performs QoS Criteria verifying process as follows. Firstly, the aggregated value of QoS Criteria, of each composite services in CS\(_{\text{set}}\) set, is computed and compared to the corresponding global QoS constraints. The composite service, which satisfies the global QoS constraints, is added to globalverfiedCS, a set of composite services that satisfy global constraints. Secondly, the Utility Value of each composite services in globalverfiedCS is computed and then the optimal composite service, which has the maximum value of utility value, is considered as the optimal composite service CS\(_{\text{maxutility_value}}\). Finally, the optimal composite service CS\(_{\text{maxutility_value}}\) is checked to know if it is the same as supposed optimal composite service CS\(_{\text{optimal}}\) by checking if the Utility Value of CS\(_{\text{maxutility_value}}\) is equal to the utility value of CS\(_{\text{optimal}}\); if they are the same, the algorithm returns true, otherwise false is returned.

4. Flight Ticket Purchase: Case Study

In this section, we show an example of how to calculate the similarity of two services by means of Algorithm 1. We provide a case study on QoS-aware composition on domain of Flight ticket purchase. In order to transform composite services from OWL-S into CPNs model, a CPNs based simulation and modeling environment called CPN Tools is used. More details about using CPN Tools is available in [19]. The following is an analysis and discussion about verification of services composition on our case study.

4.1 Scenario

The domain of our case study is Flight ticket purchase system. As a user submitted a request for flight ticket purchase, there is a need for five services, namely Flight Seat Availability service, Credit Card service, Ticket Purchase service, Email Confirmation service and Ticket Delivery service. Table 1 introduces the services and their functions.

| No | Symbol | Service Name          | Function                                         |
|----|--------|-----------------------|--------------------------------------------------|
| 1  | S\(_1\) | Flight Seat Availability | Checking available seats for user request.       |
| 2  | S\(_2\) | Credit Card           | Checking a Credit Card balance                   |
| 3  | S\(_3\) | Ticket Purchase       | Buying a certain flight ticket                   |
| 4  | S\(_4\) | Email Confirmation    | Sending an email to confirm ticket purchase      |
| 5  | S\(_5\) | Ticket Delivery       | Delivering a ticket after purchase               |

As Table 1 shows, Flight Seat Availability service S\(_1\) checks the available seats according to the submitted request of user then Credit Card service S\(_2\) checks the current balance of a credit card and if purchase procedure can be performed by current card. Ticket Purchase service S\(_3\) performs buying flight tickets for users, Email Confirmation service S\(_4\) sends emails to inform the user about purchase ticket information and Ticket Delivery service S\(_5\) delivers tickets to the address provided by the user.

The scenario of composite service can be expressed as followed: Flight Seat Availability service S\(_1\) should be invoked firstly, followed by Credit Card service S\(_2\) since only if the requested seats are
available (Choice composition construction expressed by If-Then-Else control construct in OWL-S). If
the balance of a given credit is not enough to purchase the required tickets, then Credit Card service \( S_2 \)
is invoked again to check another credit card of the user, and it should be iteratively invoked until finding
the appropriate credit card (Iterated composition construction expressed by Repeat-Until control
construct in OWL-S). Then, Ticket Purchase service \( S_3 \) will be invoked and two services Email
Confirmation service \( S_4 \) and Ticket Delivery service \( S_5 \) can be executed in parallel, the former one sends
an email to the user for confirming the transaction while the latter delivers the ticket to user (expressed
by Split-Join control construct in OWL-S). According to the scenario above and using our proposed
algebraic model for service composition, composite service can be expressed as follows:

\[
S_{\text{net}} : S_1 \Delta S_2 \Delta S_3 \Delta (S_4 \Psi S_5)
\]

(2)

where \( \Delta \) indicates a sequential operator, \( \partial \) indicates an iterative operator and \( \Psi \) denotes a parallel
operator.

4.2 Finding Composite services

As the Flight Ticket scenario is determined and composition and matching process are conducted as
shown in [20], there are many available composite services can satisfy the Flight Ticket scenario. After
filtering composite services, three composite services are selected according to the descending order of
the Utility Value of available composite services in order to apply our QoS verification as in section 3.2.

Among the three composite services, a composite service (denoted by \( S_{\text{optimal}} \)) is considered as the
optimal composite service by implementing the approach in [20]. Then, we implement Algorithm 1 to
verify QoS criteria of the three composite services, and to validate whether the optimal composite
service (denoted by \( S'_{\text{optimal}} \)), which is found by implementing ALGORITHM 1, is exactly the
determined optimal composite service (\( S_{\text{optimal}} \)) among the three composite services.

The following is the three composite services with its component services, and the values of QoS
criteria are included within brackets as the following order: price, duration, availability, reliability and
reputation.

1) Composite service \( S_1 \) includes Yiqifei(s1) -> (10,3,0,3,0,77,0,4), ICBCCreditCard(s2) ->
   \( (12,2,0,7,0,55,0,3) \), ChinaAir(s3) -> \( (3,2,0,3,0,74,0,6) \), Yahoo(s4) -> \( (2,3,0,6,0,5,0,5) \),
   EMS(s5) -> \( (2,1,0,7,0,49,0,5) \).

2) Composite service \( S_2 \) includes Elong(s1) -> \( (9,2,0,6,0,89,0,7) \), ABCCreditCard(s2) ->
   \( (11,3,0,67,0,87,0,72) \), ChinaEast(s3) -> \( (4,3,0,6,0,98,0,84) \), Gmail(s4) -> \( (1,4,0,65,0,6,0,8) \),
   SF(s5) -> \( (1,0,56,0,67,0,62) \).

3) Composite service \( S_3 \) includes QuNaEr(s1) -> \( (8,2,0,73,0,99,0,5) \), BOCCreditCard(s2) ->
   \( (9,2,0,73,0,78,0,4) \), QatarAirLine(s3) -> \( (6,2,3,0,76,0,5) \), 126(s4) -> \( (1,3,7,0,67,0,7) \),
   YunDa(s5) -> \( (1,1,0,63,0,69,0,82) \).

Among the three composite services and the result of using the approach in [20] is that \( S_{\text{optimal}} \) is
\( S_3 \). Then, using algorithm 1, we check the three composite services and validate whether the resultant
optimal composite service (\( S'_{\text{optimal}} \)) is the same \( S_3 \).

4.3 Transforming from OWL-S into CPN

Figure 4 shows a part of Flight Purchase composition model in OWL-S, which includes various types
of composition constructions, including Sequence, Parallel and Iteration.

CPN Tools is used to transfer the composition model in Figure 4 into the corresponding composition
model in CPN. Figure 5 shows the generated CPNs model of Flight Ticket Purchase.

As shown in Figure 5, all services in OWL-S are mapped to corresponding transitions, and the inputs
and outputs of the services are mapped to places associated with data types that are represented by
colored sets as shown in the left part of the figure. Control transitions, which are colored with black,
directs the flow of components of the composite services. For example, the Out1 place is checked; if
there are available flight seats, flag is true which is checked by t1, otherwise it goes through t2, which
means that a flag is false and the process is finished.
4.4 QoS aware Verification

For a given set of composite services with an optimal composite service, CS_{optimal} which is determined in advance. Verification process uses ALGORITHM 1, which computes the utility value of each
candidate composite service, and finds the one that has the maximum utility value. Finally, we check whether the resulted composite service $CS_{\text{maxutility value}}$ is exactly the same composite service that is determined in advance by checking their utility value.

Assume the QoS weights of each $w_k$, where $1 \leq k \leq 5$. Each value of aggregated QoS criteria should not violate the global QoS constraints given by the services consumer, such that: $CS_1.c_k \leq c_k$ for negative QoS criteria and $CS_1.c_k \geq c_k$ for positive QoS criteria, where $c_k$ is the aggregated QoS value of $ck$ whereas $c_k$ is the corresponding global QoS value.

QoS weights are as follows: $w_{\text{price}}=0.3$, $w_{\text{duration}}=0.3$, $w_{\text{availability}}=0.1$, $w_{\text{reliability}}=0.1$, $w_{\text{reputation}}=0.2$. Moreover, the composite global QoS value are as follows: $c_{\text{price}}=30$, $c_{\text{duration}}=20$, $c_{\text{availability}}=0.03$, $c_{\text{reliability}}=0.1$, $c_{\text{reputation}}=0.3$.

After checking the composition construction types among services, which are sequential, loop and parallel in this case study, and for the sake of simplicity we assume that the credit card service needs one loop ($l=1$) to find the suitable credit card. Composite services and their related utility value are computed as follows:

1. **Utility value of composite service $CS_1$:**
   - Aggregated value for price ($CS_1.c_{\text{price}}$) = $S_1.c_{\text{price}} \times S_2.c_{\text{price}} + S_3.c_{\text{price}} + S_4.c_{\text{price}} + S_5.c_{\text{price}} = 10+1 \times 12+3+4+2=29$ and $CS_1.c_{\text{price}}$ satisfies the global constraint as $29 \leq 30$.
   - Aggregated value for duration ($CS_1.c_{\text{duration}}$) = $S_1.c_{\text{duration}} + l \times S_2.c_{\text{duration}} + S_3.c_{\text{duration}} + S_4.c_{\text{duration}} + S_5.c_{\text{duration}} = 3+1 \times 2+2=10$ and $CS_1.c_{\text{duration}}$ satisfies the global constraint as $10 \leq 20$.
   - Aggregated value for availability ($CS_1.c_{\text{availability}}$) = $S_1.c_{\text{availability}} \times S_2.c_{\text{availability}} \times S_3.c_{\text{availability}} \times S_4.c_{\text{availability}} \times S_5.c_{\text{availability}} = 0.3 \times (1 \times 0.7) \times 0.3 \times (0.5 \times 0.49) = 0.0378$ and $CS_1.c_{\text{availability}}$ satisfies the global constraint as $0.0378 \geq 0.03$.
   - Aggregated value for reputation ($CS_1.c_{\text{reputation}}$) = $S_1.c_{\text{reputation}} + l \times S_2.c_{\text{reputation}} + S_3.c_{\text{reputation}} + S_4.c_{\text{reputation}} + S_5.c_{\text{reputation}} = 0.4+1 \times 0.3+0.6+(0.5+0.5)/2 = 0.36$ and $CS_1.c_{\text{reputation}}$ satisfies the global constraint as $0.36 \geq 0.3$.

   After computing the aggregated value for each QoS criteria, we found that none of them violate the global composite constraints.

6. $MAX_k$ denotes the maximum value of the highest QoS criteria using the corresponding aggregated function. $MAX_k = F_k^i (\max(i,k))$, where $i$ donates component services whereas $ck$ indicates a QoS criteria and $F_k^i$ denotes the composition construction types among services components. The Maximum values of each QoS criteria are computed and the result as follows: $MAX_{\text{price}}=32$, $MAX_{\text{duration}}=13$, $MAX_{\text{availability}}=0.335$, $MAX_{\text{reliability}}=0.56$, $MAX_{\text{reputation}}=0.614$

7. $MIN_k$ donates the minimum value of lowest QoS criteria using the corresponding aggregated function. $MIN_k = F_k^i (\min(i,k))$, where $i$ donates component services whereas $ck$ indicates a QoS criteria and $F_k^i$ denotes the composition construction types among services components. Minimum values of each QoS criteria are computed and the result as follows: $MIN_{\text{price}}=22$, $MIN_{\text{duration}}=9$, $MIN_{\text{availability}}=0.036$, $MIN_{\text{reliability}}=0.15$, $MIN_{\text{reputation}}=0.34$.
5. Conclusions
In this paper, a colored Petri Net based model was defined for services composition. Basically in this model, component services and their composition construction types in composite services were transferred from OWL-S to CPN using a tool called CPN tool. In addition, an algorithm was defined to verify the QoS consistency of services composition. A case study is provided to verify the result of an approach of services composition and to check the QoS consistency of services composition using the algorithm mentioned in our study.

It is clear from the above work that our approach can assist on verifying generated composite services. The results of case study obviously suggest that our study should be considered in performing verification of services composition. The method is found to be effective and efficient, due to the comprehensibility of its underlying and the straightforward computation process.

As the verification work in our approach discussed only the inconsistency of QoS criteria, future work shall involve services composition for other verification issues such as Reachability and Deadlock.

Acknowledgement
This study is partially supported by the National Key Technology R&D Program (2014BAG01B03) and the National Natural Science Foundation of China (51105286, 61203236) and Fundamental Research Funds for the Central Universities (WUT: 2014-IV-137).

References
[1] Wibowo, S.; Deng, H.; Xu, W. Evaluation of Cloud Services: A Fuzzy Multi-Criteria Group Decision Making Method. Algorithms 2016, volume 9, 84.
[2] Deng, Z.; Chen, L.; He, T.; Meng, T. A Reliability Calculation Method for Web Service Composition Using Fuzzy Reasoning Colored Petri Nets and Its Application on Supercomputing Cloud Platform. Future Internet 2016, volume 8, 47.
[3] Jinkui Hou and Lei Wang. Formal model supporting Web service composition and verification, Journal of Computer Applications 2015, volume 6, pp: 1773-1790.
[4] Manman Chen, Tian Huat Tan, Jun Sun, Yang Liu, Jun Pang, Xiaohong Li. Verification of functional and non-functional requirements of web service composition. Formal Methods and Software Engineering. Springer Berlin Heidelberg, 2013, pp. 313-328.
[5] Zahoore E, Perrin O, Godart C. Web services composition verification using satisfiability solving. Web Services (ICWS), 2012 IEEE 19th International Conference on. IEEE, 2012, pp. 242-249.
[6] Danapaquiame Nagamouttou, Ilavarasan Egambaram. A verification strategy for web services composition using enhanced stacked automata model. Springer Plus. 2015, 12(4), pp. 23-46.
[7] Jianhua Li, Songqiao Chen, Lin Jian, Hongyu Zhang. A Web Services Composition Model and Its Verification Algorithm Based on Interface Automata. 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2011, pp. 1556 – 1563.
[8] C Dumeza, M Bakhoyab, J Gabera, M Wacka, P Lorenzc. Model-driven approach supporting formal verification for web service composition protocols. Journal of network and computer applications, 2013, 36(4): pp. 1102-1115.
[9] M Emilia Cambronero, Gregorio Díaz, Valentín Valero. Validation and verification of web services choreographies by using timed automata. The Journal of Logic and Algebraic Programming, 80(1), 2011, pp. 25-49.
[10] Sycara K, Paolucci M, Anholekar A, et al. Automated discovery, interaction and composition of semantic web services. Web Semantics: Science, Services and Agents on the World Wide Web, 2011, 1(1), pp. 122-131.
[11] Yuanuyuan Zhang, Jun Liu, Qiong Tang and Yan Wu. Modeling Web Services Composition with Timed Pi Calculus. Information Technology Journal, 2011, pp. 1194-1200.
[12] Li Yongxiang, Yao Xifan, Xu Xiangmin and Jin Hong. Formal Verification of Cloud Manufacturing Service Composition and BPEL Codes Generation Based on Extended Process Calculus. Information Technology Journal, 2014, 13(11), pp. 1779-1785.

[13] Yue Ni, Yu Shan Fan. Model transformation and formal verification for Semantic Web Services Composition. Advances in Engineering Software, 2011, 41(6), pp. 879-885.

[14] Hongyue Wu, Shuxia Yu, Yuyue Du. A Logical Petri Net-based Model for Web Service Cluster Composition and Soundness Verification. Journal of Computational Information Systems, 2013, 9(13), pp. 5221–5228.

[15] Jamal Bentahar, Hamdi Yahyaoui, Melissa Kova, Zakaria Maamar. Symbolic model checking composite Web services using operational and control behaviors. Expert Systems with Applications, 2013, 40(2), pp. 508–522.

[16] Tien-Dung Cao, Patrick Felix, and Richard Castanet. WSOTF: An Automatic Testing Tool for Web Services Composition. 2010 Fifth International Conference on Internet and Web Applications and Services, 2011, pp. 7-12.

[17] Diaz Michel. Petri Nets: fundamental models, verification and applications. John Wiley & Sons, 2013.

[18] Naji Hasan, A.H, Gao Shu, AL-Gabri Malek, Jiang Zi-Long. An Optimal Semantic Network-Based Approach for Web Service Composition with QoS. TELKOMNIKA, 2013, 11(8), pp. 4505–4511.

[19] Chang Wang, Jiahe Qin, Minghua Guo, Yuanxin Xu. Prediction of Vehicle Trajectory Based on Fuzzy Colored Petri Net. Indonesian Journal of Electrical Engineering and Computer Science, 2013, 11(10), pp. 5824-5832.

[20] M. Alrifai, T. Risse, W. Nejdl. A hybrid approach for efficient web service composition with end-to-end QoS constraints. ACM Transactions on the Web, 2012, Vol 6, No 2, p. 7.