Ait-Cheik-Bihi, Wafaa; Nait-Sidi-Moh, Ahmed; Bakhouya, Mohamed; Gaber, Jaafar; Wack, Maxime

Performance Study of Workflow Patterns-Based Web Service Composition

Published in: 9th International Conference on Mobile Web Information Systems, August 27-29, 2012, Niagara Falls, Ontario, Canada

DOI: 10.1016/j.procs.2012.06.093

Published: 01/01/2012

Document Version
Publisher's PDF, also known as Version of record

Please cite the original version:

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.
Performance Study of Workflow Patterns-Based Web Service Composition

W. Ait-Cheik-Bihia, A. Nait-Sidi-Moh, M. Bakhouya, J. Gaberd, M. Wack

Abstract

Web services are currently used by organizations to share their knowledge over the network and facilitate business-to-business collaboration. However, combining Web services to satisfy user requests is a complex process. Workflow patterns are widely used for Web service composition to allow the specification of composite services. In this paper, the performance evaluation and analysis of workflow patterns is presented. We use both (max, +) algebra and Petri Nets as formal modeling tools to describe the behavior of workflow patterns and analyze their properties and performances. A real case study is worked out to represent and study the performances of integrated workflow patterns.

1. Introduction

In Service-Oriented Computing (SOC), one of the main objectives is to create value-added services (i.e. composite service) [1], which is the result of composing many services. A composite service can be modeled as a business process by using languages such as BPEL4WS or WSCI. The composite service is achieved through the invocation and coordination of its components [2], [3]. Service composition process is considered as a business process that must be made in an automatic fashion. Most of the languages used to model the business process do not allow an automated composition and still based on manual definition of new composed process that interacts with existing ones. A workflow pattern can be seen as the automatic modeling and system management for all tasks and different involved actors in a business process. In other words, a workflow pattern describes the tasks between the different actors of a process, deadlines, validation methods, and provides to each actor the information needed to fulfill its task. The use of workflow patterns has become a major asset providing a competitive advantage, as a way to reduce costs, automate processes, and reduce response times. Different studies have sought to identify the basic needs for business process modeling. The main result of these studies was the identification of a set of control structures for describing
workflow patterns, which are considered as modeling of a set of tasks performed by different actors involved in the realization of a business process. This process can be automated by invoking applications or external services and/or assigning manual tasks. In recent years much attention has been devoted to Web services composition and their modeling in the form of a business process becomes increasingly popular. In this paper, Petri net-based (PN) models and (max,+)+ algebra are combined for modeling and analyzing workflow patterns. The developed models will be used to evaluate the performance of Web service composition based on workflow patterns aggregation. We show that the complementarity of these formal tools leads to a high-level description of services composition and to the creation of a dynamic and transient relationships among workflow patterns.

2. Related work

Approaches dealing with formal modeling of Workflow patterns for service composition can be classified into two different categories, language-oriented approaches and model-oriented approaches. The former ones focus on the verification of already existing composite services expressed in a particular business process modeling language, such as BPEL or WSCI. For example, a BPEL code can be translated into a formal description, and verify some properties. The main formalisms used are Petri nets, state transition systems (STS), and process algebras. For example, in [4], the authors proposed rules for complete and accurate translation of BPEL process models to PN models. These translation rules were also widely used in other tools such as WofBPEL, which is used for automatic verification of BPEL processes.

In the second category, the designer uses graphical representations, such as BPMN or UML, to specify the service composition before generating the corresponding executable code. These business process modeling languages are widely adopted for service composition, because of the design features they provide (support most of Workflow patterns) and being comprehensive enough to generate executable codes. However, it is hard for designers to deal with formal verification of service composition because neither BPMN nor UML are supported with suitable tools. Several approaches have been proposed to take this issue by providing Workflow patterns models, and therefore designers can use them for specifying and verifying the service composition. For example, Aalst et al. proposed over forty patterns in [5]. Among them, a variety may be used to model communication and interaction between Web services. In the same context, authors in [2] defined fourteen of the twenty original patterns and used them for the composition of Web services. They also defined new patterns by composing existing ones. The new defined patterns are mainly based on the QoS as a selection criterion (e.g. cost and price ) in the composition of services. In [3], the authors defined an approach based on workflow patterns, specifically nine patterns, to adapt the composition of services during the execution of business processes. Indeed, business processes are composed of Web services that run in a volatile environment where the parameters of the QoS of participants can be changed during the execution time.

Our contribution through this paper falls into the second category by using both Petri nets and (max,+)+ algebra to specify, verify, and evaluate the performance of workflow patterns. We will use P-Timed PN wherein fixed times, called also temporizations are associated with places. These times are required for the execution of services or tasks. (max,+)+ algebra is a linear mathematical algebra enabling the modeling and analysis of certain systems whose behavior can be evolved in a discrete space. This tool has been used in the literature to evaluate the performance and control the systems mostly with discrete events [6]. We will use this algebra to manage and compute the occurrence dates of events. We refer readers to [7, 8] for more details about this algebra.

3. From Workflow patterns to (max,+)+ equations

In this section, we generate the (max,+)+ mathematical model of each considered pattern. Because we are constrained by the number of pages, we present only two potential patterns (e.g., multi-merge and conditional choice patterns) and their P-timed PN and (max,+)+ state models. Other patterns used for the Web services composition and associated models are detailed in [9]. The chosen patterns, are the most used
in the services composition process. In order to make easy the analysis of each pattern and evaluate its performances, we describe its behavior by using a \((\text{max}, +)\) state system, which can be handled and solved using the techniques of matrix calculation in \((\text{max}, +)\) linear algebra. Using a standard formalization, all patterns may be expressed under the following matrix form: \(\forall k \geq 1:\)

\[
\begin{align*}
X(k) &= A_0 \otimes X(k) \oplus A_1 \otimes X(k - 1) \oplus \ldots \oplus A_{n_0} \otimes X(k - n_0) \oplus B \otimes U(k) \\
Y(k) &= C \otimes X(k)
\end{align*}
\]  

(1)

With the marking of each place \(P\) of the PN model verifies \(m(P) \leq n_0\). The matrices \(A_0, A_1, \ldots, A_{n_0}, B\) and \(C\) are the characteristic matrices of the system whose components are the system data.

3.1. Multi-merge pattern

Multi-merge pattern is defined by the merge of two or more incoming branches in a single downstream branch, as shown in Fig. 1a. The activation of an incoming branch is enough for the activation of the downstream branch. Although several execution paths are merged without any synchronization of control flow, each active branch will therefore join the resulting merged branch. As illustrated in Fig. 1a, when one of the transitions \(x_1, x_2, \ldots, x_n\) is fired, the output transition \(y\) is activated and can be fired after the temporization \(\tau\). However, it is not obvious to formally express the firing number of the transition \(y\) according to the firing number of its upstream transitions. With the aim to describe this functioning by \((\text{max}, +)\) equations and to facilitate the mathematical analysis, we associated a place - called counter - to each one of the transitions \(x_1, x_2, \ldots, x_n\). These counters \((P_1, P_2, \ldots, P_n)\) are output places as shown in Fig. 1b. It is worth noting that the addition of these counter places will not influence the verification of certain properties of PN model, including the boundedness. Indeed, these places, whose capacity is assumed to be infinite, serve only to indicate the number of firing of the upstream transitions at a given time. Finally, these counter places do not participate in any way to change the dynamic of PN model. Using these counter places, the behavior of the modeled pattern is represented by the system (2). This gives an explicit expression of different firing dates of the transition \(y\).

\[y(k) = A_{k_1} \otimes x_1(k - k_1) \oplus A_{k_2} \otimes x_2(k - k_2) \oplus \ldots \oplus A_{k_n} \otimes x_n(k - k_n)\]

(2)

Where \(k_i = \bigotimes_{j=1, j\neq i}^{n} m_{P_j}\) for all \(i, 1 \leq i \leq n\), where \(m_{P_j}\) is the marking of the place \(P_j\), and \(A_{k_i} \in \mathbb{R}_{\text{max}}^{1 \times n}\), the \(i^{th}\) component of \(A_{k_i}\) equals to \(\tau\), and other components equal to \(\epsilon\), and the \(i^{th}\) (for \(i = 1..n\)) component of the vector \(X(k - k_i)\) is \(x_i(k - k_i)\).
3.2. Conditional choice pattern

The conditional choice pattern is represented by a conflict PN model without free choice as illustrated in the Fig. 2. In this model, if the place \( P_2 \) contains only one token, then only one of two transitions \( x_3 \) or \( x_4 \) can be fired. The conflict resolution is therefore made by choosing the transition to be fired. From a structural point of view, the transitions are validated and fired based on the availability of tokens in their upstream places. Indeed, for transitions \( x_3 \) and \( x_4 \), where \( x_3 \in P_1 \) and \( x_4 \in P_2 \) (where \( P \) is the set of downstream transitions of the place \( P \)), the firing rules are given as follows: if both places \( P_1 \) and \( P_2 \) contain a token, and the sojourn time \( \tau_1 \) and \( \tau_2 \) are expired, then the priority is given to \( x_3 \) than \( x_4 \); If the place \( P_1 \) does not contain any token, while \( P_2 \) contains one token and the sojourn time of \( \tau_2 \) is achieved, then \( x_4 \) will be fired but not \( x_3 \); If the place \( P_1 \) contains one token and \( P_2 \) does not contain any token, then neither of these transitions will be validated. In this case, the firing of \( x_3 \) depends on the arrival of a token into \( P_2 \). In order to describe the behavior of this pattern with \((\max, +)\) equations, we use the virtual firing

![Fig. 2: Conditional choice pattern](image)

(as detailed in [10]) of transitions in conflict situation. Indeed, for each firing of \( x_2 \) and after the completion of the time \( \tau_2 \), one of the two transitions \( x_3 \) or \( x_4 \) will be really fired, while the other one is assumed to be virtually fired. Furthermore, we define the two functions \( \Psi \) and \( \Psi^\prime \) by:

\[
\Psi_{a \leq b} = \begin{cases} 
\epsilon & \text{if } a \leq b \\
\epsilon & \text{otherwise}
\end{cases}
\quad \Psi_{a \leq b}^\prime = \begin{cases} 
\epsilon & \text{if } a \leq b \\
\epsilon & \text{otherwise}
\end{cases}
\]

We also define \( m_P(x_i(k)) \) as the marking of the place \( P \) at the time \( x_i(k) \). We assume that \( \forall k \leq 0, x_i(k) = \epsilon \) and the marking of a place at this time is zero. Different firing dates (including real and virtual firing) of \( x_3 \) and \( x_4 \) according to those of \( x_1 \) and \( x_2 \) are given as follows: \( \forall k \geq 1 \):

\[
\begin{align*}
x_3(k) &= [\tau_1 \otimes x_1(k) \otimes \tau_2 \otimes x_2(k)] \otimes \Psi_{\tau_1 \otimes x_1(k) \otimes \tau_2 \otimes x_2(k)} \\
&\quad \oplus [\tau_1 \otimes x_1(k - m_P(\tau_2 \otimes x_2(k) - \tau_1)) \otimes \tau_2 \otimes x_2(k)] \otimes \Psi_{m_P(\tau_2 \otimes x_2(k) - \tau_1) = 0} \\
x_4(k) &= \tau_2 \otimes x_2(k) \otimes [\Psi_{(\tau_1 \otimes x_1(k) \geq \tau_2 \otimes x_2(k)) \land (m_P(\tau_2 \otimes x_2(k) - \tau_1) = 0)]
\end{align*}
\]

The operator ’’\&’’ is defined as ’’and’’, \( a \land b \) means that both conditions \( a \) and \( b \) must be satisfied simultaneously. The terms expressing the firing dates of \( x_3 \) are given by:

- \( \tau_1 \otimes x_1(k) \otimes \tau_2 \otimes x_2(k) \) if the both following conditions are satisfied: i) \( \tau_1 \otimes x_1(k) > \tau_2 \otimes x_2(k) \) which means that a token arrives to \( P_2 \) and its sojourn time is coming to the end before the arrival of another token to \( P_1 \); ii) \( m_P(\tau_2 \otimes x_2(k) - \tau_1) = 0 \) which means that the marking of \( P_1 \) at time \( \tau_2 \otimes x_2(k) - \tau_1 \) is zero. In this case, the firing priority is given to \( x_4 \).
In the case where the transition $x_3$ (resp. $x_4$) is really fired for the $k^{th}$ time, $x_4$ (resp. $x_3$) is virtually fired for the $k^{th}$ time. The matrix form of (4) is given by:

$$X(k) = A_0(k) \otimes X(k) \oplus A_1(k) \otimes X(k - k_1)$$  \hspace{1cm} (5)$$

Where the matrices $A_0(k)$ and $A_1(k)$ are expressed as:

$$A_0(k) = \begin{pmatrix}
\epsilon & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & \epsilon & \epsilon \\
A_{31}(k) & A_{32}(k) & \epsilon & \epsilon \\
\epsilon & A_{42}(k) & \epsilon & \epsilon 
\end{pmatrix}, \quad A_1(k) = \begin{pmatrix}
\epsilon & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & \epsilon & \epsilon \\
\tilde{A}_{31}(k) & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & \epsilon & \epsilon 
\end{pmatrix}$$  \hspace{1cm} (6)$$

And:

$$\begin{align*}
A_{31}(k) &= \tau_1 \otimes \Psi_{\tau_1 \otimes x_3(k) > \tau_2 \otimes x_2(k)} \\
A_{32}(k) &= \tau_2 \otimes \left[ \Psi_{\tau_1 \otimes x_3(k) > \tau_2 \otimes x_2(k)} \oplus \Psi_{m_{P_1}(\tau_2 \otimes x_2(k) - \tau_1) \neq 0} \right] \\
A_{42}(k) &= \tau_2 \otimes \Psi_{\tau_1 \otimes x_3(k) > \tau_2 \otimes x_2(k) \wedge (m_{P_1}(\tau_2 \otimes x_2(k) - \tau_1) = 0)} \\
\tilde{A}_{31}(k) &= \tau_1 \otimes \Psi_{m_{P_1}(\tau_2 \otimes x_2(k) - \tau_1) \neq 0} \\
\text{with } k_1 &= m_{P_1}(\tau_2 \otimes x_2(k) - \tau_1)
\end{align*}$$

The system (5) is a non-stationary system whose matrices depend on the parameter $k$. This aspect of non-stationarity is due to the conflicts depicted in the associated PN model of the Fig.2.

4. Case study

In this section, a service composition scenario is proposed to illustrate the practical use of workflow patterns and $(\max, +)$ algebra. In this scenario, several road-related services are composed and interact with each other to minimize the emergency response time in the case of dangerous situations (e.g. accident, fire).

4.1. Scenario description

The scenario used in our case study consists of four Web services that interact (see Fig. 3) to fulfill the driver query when getting into an emergency situation. These services are described in the following.

Fig. 3: The studied scenario

The **Permanence** service is invoked by the user from its device [11] providing the incident location and the incident type. The request is then submitted to other services such as Police service or hospital service depending on the nature of the incident (e.g. in case of accident Police, Ambulance, and Hospital services are invoked). The **Police** service receives from Permanence service the location of the incident and contacts the nearest available police station to reach the incident location. This service use a cooperative plateforme
Table 1: Legend of the PN scenario model

| P1  | Permanence WS            |
| T1  | Informing police service in case of no dangerous incident |
| T2  | Informing all services in case of dangerous incident |
| P2  | Researching the nearest police station |
| T3  | Provides the police location |
| P3  | Researching the nearest emergency location |
| T4  | Provides the ambulance location |
| P4  | Researching the nearest hospital location with available service |
| T5  | Provides the hospital location |
| P5  | Asking for an itinerary from the current location of the police to incident location |
| T6  | Provides the computed itinerary |
| P6  | Asking for an itinerary from the ambulance location to incident location |
| T7  | Provides the computed itinerary A |
| P7  | Waiting for hospital response |
| P8  | Waiting for ambulance response |
| P9  | The traveled time of the police |
| P10 | The traveled time of the ambulance to incident location |
| P11 | Asking for an itinerary from the incident location to hospital location |
| T9  | Provides the computed itinerary B |
| P12 | The traveled time of the ambulance from incident location to hospital location |
| T10 | Provides all responses |
| P13 | The final response of the request |
| y   | Sending the final response to the requester |

for road security, called TransportML and developed in [12], to compute an itinerary from their current location to the incident location. The Emergency service receives the incident location and then looks for the nearest emergency station for intervention. The computed shortest path to reach this destination by using TransportML platform is provided to the service. Finally, the Hospital service receives the incident location and provides the nearest hospital to the incident location with available hospital beds. The information about the available hospital and its location is sent then to the Ambulance service.

4.2. Scenario modeling

The P-timed PN model of the considered scenario is given in Fig. 4. In this PN model, we identify the following patterns: sequence \( (T_4, T_7, \text{and } T_{10}) \), multi-merge \( (T_1, T_2, \text{and } T_3) \), differed choice \( (U, T_1, \text{and } T_2) \), synchronizer \( (T_4, T_5, \text{and } T_8) \), and parallel split \( (T_2, T_3, T_4, \text{and } T_5) \). The counter places \( Q_1, Q_2, Q_3, \text{and } Q_4 \) are added as described in section 3.1. Table 1 describes the meaning of the different elements of the PN model.

Fig. 4: The PN model of the scenario
To obtain the mathematical model of the P-timed PN described in Fig. 4, we associate with each transition $T_i$ a date $x_i(k) \in R_{max}$ which means the occurrence date of the event $T_i$ for the $k^{th}$ time. We provide then the $(max,+)$ equation of each identified pattern. Once all the $(max,+)$ equations are acquired, they are merged to obtain the general $(max,+)$ system given by: $\forall k > k_i, k > k_j$:

$$\begin{align*}
X(k) &= F(k) \otimes A \otimes X(k) \oplus A_{k_i} \otimes X(k - k_j) \oplus F(k) \otimes B \otimes U(k) \\
Y(k) &= C_{k_i} \otimes X(k - k_j)
\end{align*}$$

(7)

Where: $X(k) \in R_{max}^{1 \times 1}$ (resp. $U(k) \in R_{max}^{1 \times 1}$ and $Y(k) \in R_{max}^{1 \times 1}$), is the state vector (resp. the input and output vector) and the $i^{th}$ component of $X(k)$ is $x_i(k)$. The characteristics of the other elements of the equations 7 are provided as follows:

- $F(k) \in R_{max}^{11 \times 11}$ is the routing matrix that gives the firing rules of transitions in conflict (more details about this matrix is given in [10]). This matrix is defined as follows: $\forall i, j \in \{1, ..., 11\}$ where $i \neq j$, $(F(k))_{ij} = \epsilon$. For $i = j$, we have $(F(k))_{2,2} = (F(k))_{4,4} = (F(k))_{5,5} = (F(k))_{7,7} = (F(k))_{8,8} = (F(k))_{9,9} = (F(k))_{10,10} = f_{22}(k)$, $(F(k))_{1,1} = (F(k))_{11,11} = f_{11}(k)$, and $(F(k))_{3,3} = (F(k))_{6,6} = f_{11}(k) \otimes f_{22}(k)$.

- $B \in R_{max}^{1 \times 1}$, where $B' = (\tau_1, \tau_1, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon)$

- $A_{ki} \in R_{max}^{11 \times 11}$ and: $\forall j, l \in \{1, ..., 11\}, A_{ki,l} = \epsilon$ except when $j = 3$ and $l = 1$ ; $(A_{ki,l})_{3,1} = \tau_2$ and when $j = 3$ and $l = 2$ ; $(A_{ki,l})_{3,2} = \tau_2$.

- $C_{ki} \in R_{max}^{1 \times 1}$ where:

- $C_{k_{10}} = (\epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon)$ and $C_{k_{11}} = (\epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon)$

- $\forall i \in \{3, ..., 11\}, k_i = \epsilon$ except for $k_1 = m(Q_2)$ and $k_2 = m(Q_1)$

- $\forall l \in \{1, ..., 9\}, k_l = \epsilon$ except for $k_{10} = m(Q_4)$ and $k_{11} = m(Q_3)$

- $A \in R_{max}^{11 \times 11}$; $A = \begin{pmatrix}
\epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & \tau_2 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & \tau_3 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & \tau_4 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & \epsilon & \tau_5 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & \epsilon & \epsilon & \tau_6 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & \epsilon & \tau_6 & \tau_8 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \tau_10 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \tau_9 & \tau_10 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \tau_9 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon
\end{pmatrix}$

The $(max,+)$ equation (7) can be solved and the response time required to answer the user requests can be then analytically evaluated.

4.3. Scenario implementation and tests

The scenario is developed and implemented using Java/J2EE and eclipse IDE. The JAX API was used to implement the Web services involved in the considered scenario. Ambulance and Police services request TransportML broker to compute an itinerary avoiding areas where congestion or road not cleaned from snow [12]. Data required for formal modeling of the scenario, such as the numerical values of different temporizations and different firing times of the input transition $U$ are gathered from experimental modeling by taking the average values. Fig. 5 illustrates the simulation and experiment results. Fig. 5a shows that the response time increases and decreases in a periodic way. These periodic variations are due to the routing policy proposed to manage the appeared conflicts in places $P_1$ and $P_9$ (see Fig.4). As stated previously,
the permanence service invokes Police, Ambulance and Hospital services depending on the nature of the incident. In the case of dangerous incidents all services are invoked, otherwise only Police service is invoked. In Fig. 5b, the response time is around (5s) comparing to other executions. Indeed, during the first call of services there is a process of service discovery, the invoked services are then serialized for subsequent calls. From the 4th call of services, the response time is improved and remains constant around (1s).

5. Conclusion

In this paper, Petri net and (max,+) algebra are used for describing workflow patterns involved in Web service composition. Mapping rules from Petri net models of workflow patterns to (max,+) algebra equations are provided. A real case study is illustrated to show the practical use and the efficiency of (max,+) algebra for specifying and evaluating the performance of integrated workflow patterns. Ongoing work includes the modeling and the analysis of concrete scenarios of Web services composition integrating other workflow patterns. In addition, various QoS criteria will be taken into account and integrated in the formal models in order to ensure a better selection of services with best scores of QoS. Further experiments and tests will be conducted including different metrics of evaluation.

References