Towards Validation of E-Services Composition: Transforming eFlow to Petri-Nets

H. Nematzadeh¹, and R. Mohamad¹
¹Faculty of Computer Sciences & Information Systems, Universiti Teknologi Malaysia, Skudai, Johor, Malaysia

Abstract- Web services composition is the ability to integrate existing services to have value added services. Workflow-based services composition is one of the approaches proposed. eFlow is one of the Workflow-based service composition approaches. The approach is strong in providing a platform for adaptive and dynamic web services composition. However, it neglects the validation of the Quality of Services (QoS). This paper proposes an initial work to validate the QoS of web services composition. We propose an approach to transform eFlow to Petri-Net. We also extend existing Petri-Net by introducing a new component with new firing rule. Our preliminary result is to show that formal validation techniques like Petri-Net are useful towards validating the QoS of e-services composition.

Keywords: eFlow, Petri-Net, Validation, QoS

1 Introduction

Service is a small and distinct unit of logic. Collectively these units comprise a larger piece of business automation logic. Individually these units can be distributed. E-services, internet services, are among the fundamental concepts in Service Oriented Architecture [1]. In order to have value-added and integrated services, it is needed to compose existing e-services. There are different ways of composing e-services. Several web-services composition approach have been introduced in [2]. Systems are created by reusing services available and dynamically binding service requests with the available services and resources in a given context. Hence, quality of services (QoS) of SOA plays an important role in enabling dynamic interoperation and collaboration among service providers, service brokers, and service consumers. Moreover web service providers should satisfy web service users’ non functional conditions to survive in today competitive market. In this respect, validating the QoS of SOA is important and poses significant challenge due to its dynamic and unpredictable nature. Formal methods have wide practicality to validate the specification of a system especially when the system is complex. They can help developers and designers to obtain either functional or non-functional properties of a system.

Many approaches of web services composition are proposed. Workflow-based is one of the approaches. eFlow [3] is one of the workflow-based composition methods that consider a dynamic workflow generation. It models a composite service by the graph that defines the order of execution among the nodes in the process. The composite services can dynamically modify service process definition especially in a case where user interventions are required. However, eFlow neglects specification of nonfunctional QoS properties such as reliability, security, dependability, or scalability [2],[9] and support for validating the Quality of Services (QoS). In order to allow for formal validation of the QoS e-services composition, we extend the eFlow modeling framework into Petri-net based models. The choice of Petri-net is due to the following reasons:

1) Understandability: Petri-Net as state based model is graphical and formal. The graphical concept of Petri-Net helps better understandability of the system in terms of flow of works.

2) Formality: Petri-Net able to formulate the problem and find new solutions to difficult questions using Petri-Net properties with the help of mathematical foundations.

3) Expressiveness: Petri-Nets support all the primitives needed to model a workflow process.

4) Commonality: Petri-Net and eFlow have the concept of workflow in common. eFlow has the concept of service composition as a work flow and literature shows that Petri-Net and its extensions are suitable for work flow systems.
Literature shows that Petri-Net is applicable and useful in field of workflow systems [10,12,13] as well as service composition[11]. The paper is structured as follows. In Section 2, we present the eFlow modeling framework. Section 3 presents an overview of Petri-Net. Section 4 presents a transformation of eFlow to Petri-Net. Section 5 describes how Petri-Net is useful to find the quality values of eFlow. Section 6 describes case studies to illustrate the applicability of the model. Finally, Section 7 provides some comparisons among our research and related works and section 8 presents some concluding remarks and discusses future research directions.

2 eFlow

eFlow is a system that supports the specification, enactment, and management of composite e-services, modeled as processes that are enacted by a service process engine [3]. In eFlow, a composite service is described as a process schema that composes other basic or composite services. Figure 1 illustrates a process definition with three basic services and one composite service. A service process instance is an enactment of a process schema. The same service process may be instantiated several times, and several instances may be concurrently running.

3 Petri-Net

Petri-Net belongs to the group of state based model category. These models use a control flow graph to represent the system architecture. A Petri-Net is one of several mathematical representations of discrete distributed systems. As a modeling language, it graphically depicts the structure of a distributed system as a directed bipartite graph with annotations. As such, a Petri-Net has place nodes, transition nodes, and directed arcs connecting places with transitions.

A Petri-Net is a 5-tuple($S$, $T$, $F$, $M_0$, $W$) [4], where $S$ is a set of places. Places are shown using circles inside Petri-Net. $T$ is a set of transitions which are depicted using rectangular bars, and $F$ is a set of arcs known as a flow relation. Arcs are shown using arrows. The set $F$ is subject to the constraint that no arc may connect two places or two transitions, or more formally as in equation 1:

$$F \subseteq (S \times T) \cup (T \times S)$$

$M_0 : S \rightarrow \mathbb{N}$ is an initial marking, where for each place $s \in S$, there are $n_s \in \mathbb{N}$ tokens. $W : F \rightarrow \mathbb{N}$ is a set of arc weights, which assigns to each arc $f \in F$ some $n_f \in \mathbb{N}$ denoting how many tokens are consumed from a place by a transition, or alternatively, how many tokens are produced by a transition and put into each place. Besides the formality of Petri-Net which was mentioned above the graphical view of Petri-Net is another power of Petri-Net.

<table>
<thead>
<tr>
<th>Petri-Net components</th>
<th>Graphically Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Token</td>
<td>●</td>
</tr>
<tr>
<td>Transition</td>
<td></td>
</tr>
<tr>
<td>Place</td>
<td>○</td>
</tr>
<tr>
<td>Arc</td>
<td>→</td>
</tr>
</tbody>
</table>

Table 1. Components of Petri-Nets

Figure 1. Sample process definition
4 Transformation of eFlow to Petri-Nets

The transformation begins with mapping the starting point of the eFlow to a source transition in Petri-Net. The source transition is responsible for generating tokens inside the system. Also the ending point is a place in a Petri-Net. This place is for collecting processed tokens. The next step of mapping consists of mapping each service of eFlow (either basic or composite) to a relevant Petri-Net. Each single service can be mapped to a single transition in Petri-Net. Sometimes services are sequential; the second service won’t be done unless the first service finished. In order to map sequential services a place is put between two transitions, each for one specific service. Moreover the eFlow has the concept of decision nodes to specify the alternatives and rules controlling the execution flow. And split/join and Xor split/join will be mapped using the properties of simple Petri-Net. Xor split/join is responsible for checking the loop condition at the bottom. The reset arc links the Xor split to Xor join. For event nodes, a transition can map either notify or request event. Just for simplicity in representation and distinguishing notify event from request event, we added “N” and “R” label to the relevant transitions. Firing rule for event nodes follows the ordinary Petri-Net. The multiservice nodes allow the multiple, parallel activation of the same service node in eflow. To map the multiservice node we use either Xor split/join or And split/join based on the condition and design of eflow. Again for simplicity in presentation we added “M” label to the relevant transitions to distinguish them from ordinary Xor split/join or And split/join. Table 2 shows the transformation table.

Table 2. Transformation table from eFlow to Petri-Net

<table>
<thead>
<tr>
<th>Concept</th>
<th>eFlow</th>
<th>Petri-Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting Point</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>Ending Point</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>Single Basic or Composite Service</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
</tbody>
</table>

4.1 Enhancement of Petri-Net

A new kind of Petri-Nets transition is introduced for a generic node of eFlow. In contrast to simple transition in Petri-Net, generic node transition has a new enhanced firing rule. This is because we have an explicit view for constructing Petri-Net in our model:

- A generic node split is said to be enabled iff each input place p of generic node split contains at least one token and
• Based on the routing rules inside Generic node split, output arcs should be fired. (Generic node route can fire more than one output arc. If transition condition holds, then the transition necessarily doesn’t produce a token in each output place of generic node split.)

To explain more about generic nodes firing, suppose there are three services in service node pool. The generic node split can skip any of these three services using a boolean variable like neededservice. Based on generic node definition at least one of the transitions’(t1,t2,t3) neededservice should be true. Figure 2 shows the schema of new defined node and skip conditions.

4.2 Transformation of eFlow error events

Sometimes services fail and won’t complete because of time limitation, server crash and other reasons. Based on a low level behaviour of the task in Petri-Net as in Figure 3, we can map error events as in Figure 4. In Figure 3, the token never passes transition T unless transition T commits it. If T doesn’t commit, the token will be returned to place P1 through rollback transition. In Figure 4, a transition is responsible for failure occurrence. If rollback transition fires in Figure 3, then Transition Failure (TF) will fire in Figure 4. Likewise if commit transition fires in Figure 3, then T will fire in Figure 4.

5 Quality values with Petri-Net and WSLA modification

One way to show how we can find quality values through Petri-Net is to attach probability of firing to each transition. Probability of firing is the percentage of commitment in figure 4. If transition T in figure 4 commits 0.95 then transition TF fails 0.05 of total firings. In figure 5 a portion of Petri-Net is provided. Assume the token in the input place of Xor split (i) has the probability of firing of n = 0.95. We want to show how we can find the probability of firing in the output place of Xor join (o) through Petri-Net deterministically. Through testing we know that the route with transition t1 has the occurrence probability of op1 = 0.6 and the route with transition t2 has the occurrence probability of op2 = 0.4. (op1 + op2) = 1.

Figure 2. Generic node transition and skip conditions

Figure 3. The low level behaviour of the transition T

Figure 4. Service failure occurrence in eFlow
The quality value which here is the probability of firing in the output place of Xor join, o, will be calculated as follow:

\[
\text{Quality value at } o = n \times \left( (op1 \times c1) + (op2 \times c2) \right) \\
\Rightarrow 0.95 \times \left( (0.6 \times 0.95) + (0.4 \times 0.97) \right) = 0.9101
\]

In this case simply the quality value (qv) of a token when it passes a transition with commitment ratio of c with the occurrence probability of op is provided in equation 2:

\[
(qv)_{\text{old}} \times (op \times c) = (qv)_{\text{new}} \quad (2)
\]

The concept of commitment ratio (probability of firing) is helpful to find the reliability of eFlow. One way to find the measurable qualities like the probability of firing which are needed to calculate the certain quality value is to find and attach them in Web Service Level Agreement (WSLA) [14]. The motivation is because QoS requirements and QoS capabilities are in WSLA. Therefore whenever a measurable quality of a service is needed for a certain quality value to be calculated in eFlow, service users can refer to WSLA and approach the certain measurable quality. WSLA can be either in text document or XML.

6 Illustrative examples

In this section, we show examples how the eFlow model is transformed to the relevant Petri-Net model. We also discuss some behavioural and structural evaluations on the Petri-Net models.

6.1 Candidate interview service

Figure 6 shows a simple graph describing a composite service that helps users in organizing a net base interview offered by a company to simplify candidate interviewing [5]. The composite service includes simple services provided by the company itself (such as Interview Room Reservation) as well as services provided by partners such as Immigration Status Analysis which is offered by law firm. Sequential services, And join/split, Xor split/join, notify event node, request event node and multiservice node are used in this eFlow. Figure 7 illustrates the relevant Petri-Net model of the Candidate interview eFlow model.

6.2 Customove

Figure 8 presents the sample of customove process and shows how the process is instantiated according to the customer’s input [6]. In the scenario, customers specify the services they need by accessing the customove web page and by selecting the checkboxes corresponding to the required services. As the customer submits the form, a new instance of the customove process is started, and the list of selected services is passed as part of the process instance input parameter, in order to configure the generic service node Furniture Moving Services. The new instance will be executed according to the customove process definition where the generic service node has been replaced by a set of service nodes in eFlow (whose definition is loaded from the service node repository) to be executed in parallel, according to

![Figure 5. Quality value at place O](image)

![Figure 6. Candidate interview eFlow](image)
6.3 Behavioural analysis

This section discusses how the tokens move inside the Petri-Net for each scenario. \( t_0 \) in both two Petri-Nets is responsible to generate new tokens inside the system. This happens when the user or customer opens a relevant webpage to start a new query. In each of the Petri-Nets, exactly one place is labelled with \( "i" \) (input place), at least one \( "o" (o_1...o_n) \) (output place for successful scenarios) and exactly one \( of \) (output place for unsuccessful scenarios). Notice that the number of \( o \)'s in Petri-Net matches the number of ending points in eFlow. For candidate interview service, \( t_1 \) fires when the customer enters all data correctly. \( t_2 \) fires if the nationality of the customer is not American. \( t_3 \) fires when the interview schedule definition is done. \( t_4, t_5 \) and \( t_6 \) should be fired in parallel since there are between And split/join. Then, \( t_7 \) requests an event from eFlow engine. \( t_8 \) and \( t_9 \) are same service node transitions, activated twice to issue interview results (multiservice node). In this example we assumed both of the team member should issue the result therefore we used And split/join to show the multiservice node. \( t_{10} \) fires when get final evaluation completes successfully. \( t_{11} \) is a notify event node that notifies eFlow engine which is nonblocking and failure is not assumed for this transition. \( t_{11} \) is a nonblocking transition however \( t_7 \) spends time. Black transitions are \( tf \), responsible for finding failures and passing the tokens to the \( of \).

For customove eFlow, \( t_1 \) fires when data is collected by the customer. Then, all the services inside the generic node are represented between generic node split/join statically through 5 transitions \( (t_2,...,t_6) \). However to select the number of arcs that link the token to the relevant transition, generic node split follows the routing rule inside itself. In the customove webpage, there is a checkbox that customers select the services they want. In our model we assumed the generic node execution mode is parallel. Finally, \( t_7 \) fires when billing is completed. All processed tokens will be collected in place \( o \). Again black transitions (\( tf \)) are responsible for finding failures and passing the tokens to the \( of \).
6.4 Structural analysis

First, we restrict our model to arcs with weight one. Our enhanced Petri-Net is bounded and safe, since the number of tokens in each place does not exceed one. Furthermore, all of the generated Petri-Nets will have one initial place, which activates the Petri-Net when it is marked (i), and at least two exit places(o, off), which do not have any postconditions and cannot be marked simultaneously. These exit places correspond to the correct or erroneous termination of the system represented by the Petri-Net. For any cases, either successful or unsuccessful, the Petri-Net will terminate eventually (rule 1) and the moment the Petri-Net terminates there is a token in one of the output places (o, on, or of) and all the other places are empty (rule 2). Rule one and two show the concepts known as proper termination.

7 Related work

Our proposal complements analysis techniques proposed by Zhong et al [7], H.Kang et al [8]. In [7], Zhong et al used implicit way of modelling, however we used explicit way of modelling where the moment of choice is sooner than implicit modelling. Moreover, we propose a new model to handle failure occurrence. We specifically focused on eFlow for its strength in dynamic web-services composition. Existing Petri-Net is enhanced to handle the eFlow models like event nodes, generic nodes and multiservice nodes, each of them with different specifications.

In [8], H.Kang et al provided a CPN modelling for web service compositions in general and they’ve done some analysis. We’ve also done structural analysis on our model however we scrutinized the concepts of service composition in more detail by checking eFlow closely. Likewise there are already other researches in the area of QoS and web service composition. In [15], Y. Chen et al first proposed a new service description language, then proposed a six step composition selection algorithm. There is another research very close to our research since it applied Petri-Net to build reliable web service [16] in which P. Chan et al proposed a dynamic Web service composition algorithm that improved reliability of web service. The verification of composed web services was done using Petri-Net. P.Xiong et al [17] built a web service configuration net based on Petri nets in order to exhibit web service configuration in a formal way. Our proposal transforms eFlow to Petri-Net. The Petri-Net model is evaluated behaviourally and structurally. The advantages of doing this transformation is to use the formalism of Petri-Net as it is shown in section 5 and find quality values in terms of measurable values.

8 Conclusions and future work

This paper discussed transformation of eFlow modeling framework to Petri-Net. The ordinary Petri-Net is enhanced to include a firing rule of a new defined component (generic split/join). It has been also presented that how Petri-Nets help to find out quality values of eFlow. Furthermore, it has been discussed that it could be better to find the measurable qualities and attach them in WSLA. Then two case studies, Candidate interview service and Customove, were analyzed and mapped to Petri-Net based on the transformation model. At last, the behavioural and structural specifications of the Petri-Net were analyzed. The novelty of our approach lies in new Petri-Net model for eFlow as well as providing behavioural and structural analysis. Likewise to explicit modelling of eFlow, we identified a new component to our Petri-Net model with new firing rule. The contribution of our preliminary results is to show that formal evaluation techniques like Petri-Net are useful towards validating the QoS of e-services composition.
There are several directions for future research. First, we are working on extending the proposed model to include some non-functional parameters like reliability, performance, availability and etc. We are also working in extending the proposed model to include all workflow-based characteristics so that a holistic QoS validation model for e-services composition can be proposed. Second, we will modify WSLA and add the concept of new measurable qualities in machine readable, XML format, to find measurable qualities easier for service users to calculate QoS in e-services composition. Third, we are working on a CASE tool to simulate the transformation and validating the QoS for the e-services composition.

References


