A Petri Net Approach to Mediation-Aided Composition of Web Services
Yanhua Du, Xitong Li, and PengCheng Xiong

Abstract—Recently, mediation-aided composition has been widely adopted when dealing with incompatibilities of services. However, existing approaches suffer from state space explosion in compatibility verification and cannot automatically generate the BPEL code. This paper presents a Petri net approach to mediation-aided composition of Web services. First, services are modeled as open WorkFlow Nets (oWFNs) and are composed using mediation transitions (MTs). Second, the modular reachability graph (MRG) of composition is automatically constructed and the compatibility is analyzed, so that the problem of state space explosion is significantly alleviated. Furthermore, an Event-Condition-Action (ECA) rule-based technique is developed to automatically generate the BPEL code of the composition, which can significantly save the time and labor of designers. Finally, the prototype system has been developed.

Note to Practitioners—Web services are an emerging area for business process automation. This work presents a novel Petri net approach to mediation-aided composition of Web services. The proposed approach can greatly alleviate state space explosion to automatically verify the composition of partially incompatible services, and significantly save the time and labor of designers to obtain BPEL code. It consists of three phases: modeling composition of Web services, automatic verification of composition, and automatic generation of BPEL code. The prototype system has been developed based on the open source software PIPE and validated in a real-life case study. It is ready to be applied in industrial Web service composition for business automation.

Index Terms—Compatibility verification, mediation transition, mediation-aided composition, modular reachability graph, prototype system.

I. INTRODUCTION

Service-Oriented Architecture (SOA) is becoming one of the main computing paradigms for designing complex business applications [1], [2]. Usually, a business application is not realized by a single Web service but a set of them. Composition, in which multiple independent Web services are assembled to accomplish a more complex task, is one of the key motivations to embrace Web service technology [2]–[6].

According to whether or not the participating Web services (abbreviated to services in the rest of this paper) in the composition are exactly compatible, service composition can be divided into direct composition and mediation-aided one.

Various direct composition methods have been proposed, including planning based [1], logical inference driven [2], Petri net based [3], [4], automata based [5], quality-of-service (QoS) optimizing based [6], etc. These methods [1]–[6] assume both data formats and sequences of the messages are consistent. However, services are not always exactly compatible in real-life composition situations. Usually, two (or more) services providing complementary functionality could be linked together in principle, but cannot be directly composed because of partially compatible interfaces or interaction patterns.

Mediation-aided composition [7]–[15] is attracting more attention, which mainly uses a set of mediators/adaptors to glue two or more partially compatible services. Compatibility verification is a crucial task of mediation-aided composition which is used to check whether there exist mediators to glue two partially compatible services [7]–[15]. Another important task of mediation-aided composition is to automatically generate the (abstract) BPEL code of composition, since BPEL has become the industrial standard for modeling service composition. This can significantly save the time and labor of designers with the fast changing need.

Existing work [7]–[15] has not fully investigated the issue of mediation-aided composition, because they suffer from state space explosion in compatibility verification and cannot automatically generate the (abstract) BPEL code. In this paper, a Petri net approach to mediation-aided composition of services is presented. First, services are modeled using open WorkFlow Nets (oWFNs) [16], and are composed by adding mediation transitions (MTs). Second, the composition compatibility is verified by automatically constructing and analyzing the modular reachability graph (MRG) [17] of composition. Finally, if the composition is verified to be valid, the BPEL code of the composition is automatically generated in an Event-Condition-Action (ECA) rule-based way [18].

Compared with the existing work [7]–[15], the contributions of this paper are as follows.

1) oWFNs of services are composed by using three basic kinds of MTs to address the problem of their partially compatible interfaces or interaction patterns.
2) By automatically constructing and analyzing the MRG of composition, our approach can significantly alleviate state space explosion without unfolding to ordinary state space.
3) Once the composition is verified to be valid, its BPEL code is automatically generated in the format of ECA rule, which can significantly save the time and labor of designers with the fast changing need.
4) The prototype system based on the open source software Platform Independent Petri net Editor (PIPE) has been developed.

Note that our approach is an offline one. Once the composition is launched, no runtime reconfigurations are possible, e.g., services cannot be replaced or reconfigured during execution, because our approach assumes that the message mappings among services to be composed are specified by designers and should be accurate and faultless.

The rest of this paper is organized as follows. Section II presents the composition of oWFNs by adding MTs. Section III presents how to automatically construct the MRG and analyze its compatibility. Section IV discusses automatically generation BPEL code from composition. Section V presents a prototype system. Section VI discusses related work and Section VII concludes this paper.

II. MODELING MEDIATION-AIDED COMPOSITION OF SERVICES

In this section, first, the formal concept of open WorkFlow Net (oWFN) is introduced. Second, mediation transitions (MTs) between oWFNs are presented. Finally, the composition procedure of oWFNs based on MTs is presented.
### A. Open Workflow Net (oWFN)

Existing service composition specification languages such as Business Process Execution Language (BPEL), Web Service Choreography Interface (WSCI), and Web Service Choreography Description Language (WS-CDL) all provide mechanisms to compose services by specifying message sent or received by interfaces [9]. Among these various languages, BPEL has become dominant because it has been proposed by OASIS as an industry standard [7]–[10] and is supported by major software companies such as IBM, Oracle, and SAP. In this paper, BPEL is assumed as the language for describing the internal logic of services and the final composition of them. By doing so, our theoretical approach becomes practical and can be used to address real-world services in practice.

To formally analyze the composition capability of BPEL services, we first model them based on oWFN. As a special class of Petri nets [7]–[10], [19], oWFN [16] is generalized from the classical Workflow Net (WFN) [7]–[10] by introducing the interfaces for exchanging messages.

**Definition 1. (Open Workflow Net, oWFN):** An oWFN is a 6-tuple \( \alpha = (P, T, F, I P, O P, F') \) where:

1. \( \{ P, T, F \} \) is a WFN;
2. \( IP \) is the set of input message places, and \( \forall x \in IP, x = \emptyset \);
3. \( OP \) is the set of output message places, and \( \forall x \in OP, x^* = \emptyset \);
4. \( F' \subseteq (T \times OP) \cup (IP \times T) \) is the set of interface arcs.

Assume there are two BPEL services to be composed: eShop service and a Third-Party Checkout service (TPC), which is the excerpt and a fragment of the resource service. The two services are defined as follows.

- **eShop Service:** A service that represents the process of purchasing items and checking out. It includes the following places:
  - \( p_1 \) (Place): Waiting for a message.
  - \( p_2 \) (Place): Waiting for a message.
  - \( p_3 \) (Place): Waiting for a message.
  - \( p_4 \) (Place): Waiting for a message.
  - \( p_5 \) (Place): Waiting for a message.
  - \( p_6 \) (Place): Waiting for a message.
- **TPC Service:** A service that represents the process of processing the payment and shipping information. It includes the following places:
  - \( p_1 \) (Place): Waiting for a message.
  - \( p_2 \) (Place): Waiting for a message.
  - \( p_3 \) (Place): Waiting for a message.
  - \( p_4 \) (Place): Waiting for a message.
  - \( p_5 \) (Place): Waiting for a message.
  - \( p_6 \) (Place): Waiting for a message.

When buyers check out: 1) eShop service is initiated, and it invokes TPC service by sending message OrderData. 2) eShop service receives messages including data CardToken and ResSecID from TPC service, and sends synchronously data CardID and Password to TPC service as a message. 3) eShop service receives message composed of ResSecID, OrderID and UserID from TPC service, and replies asynchronously an optional message OrderData.

### B. Mediation Transition

Based on the Web Service Description Language (WSDL) specifications of messages exchanged of oWFNs, the message mappings between two them can be set.

**Definition 2. (Message Mapping, MM):** A message mapping MM between two oWFNs is expressed in the form of \(<source, target>\), where source is the messages, or their parts/elements that need to be sent by an oWFN, and target is the messages, or their parts/elements that need to be receive by another one.

Source and target are expressed in the form of Service.Message or Service.Message.Part. In this paper, it is assumed that the MM services to be composed are specified by designers and the MM should be accurate and faultless. The automatic generation of MM is beyond the scope of this paper.

For superfluous message pattern, the MM between two oWFNs is \(<source, \emptyset>\). The MM of missing message pattern is \(<\emptyset, target>\). These patterns cannot affect the verification result of service composition and do not appear explicitly in the composition models. Superfluous messages can be discarded by designers, because they do not lead to deadlocks of the oWFNs. On the other hand, if the designers cannot provide the missing messages, then the composition is usually considered to be incompatible and do not need further verification.

**Definition 3. (Message Place Mapping, MPM):** A message place mapping MPM is transformed from a MM \(<source, target>\) and expressed in the form of \(<ps, ps'>\), where \(ps\) is the set of output message places corresponding to messages (or their parts/elements) in source and \(ps'\) is the set of input message places corresponding to messages (or their parts/elements) in target.

Not all of input or output message places of oWFNs need to appear in a MPM. For these redundant output messages that no oWFN accepts, their corresponding places will not appear in a MPM. For an oWFN with choice branches, if one path is not picked up by its partner oWFN, the corresponding input message place will also not appear.

Based on MPMs, we can derive mediation transitions to connect oWFNs, which serve as information channel by specifying the transferring relation of messages.

**Definition 4. (Mediation Transition, MT):** A mediation transition MT is a transition which has at least one input places and at least one output places in oWFNs.

In this paper, three basic kinds of MTs are defined as follows.

1. **Forward Mediation Transition (FMT):** A FMT stores the incoming message and forwards it to the receiver when needed. For a MPM \(<ps, ps'>\), where \(ps\) and \(ps'\) all have only one message place, a FMT is used to connect the output message place in \(ps\), and the input message place in \(ps'\).
2. **Merge Mediation Transition (MMT):** A MMT collects the multiple source messages/ports/elements and then combines them into one single target message. For a MPM \(<ps, ps'>\), where \(ps\) has more than one message places and \(ps'\) has only one, a MMT is used to connect the output message places in \(ps\), and the input message place in \(ps'\).
3. **Split Mediation Transition (SMT):** A SMT replicates the source message (or its part/element) into multiple copies. For a MPM \(<ps, ps'>\) where \(ps\) has only one message place and \(ps'\) has more than one, a SMT is used to connect the output message place in \(ps\), and the input message places of in \(ps'\).

**Definition 5. (Composition of Open Workflow Nets by MTs, CoWFN):** Suppose \(oWFN_{i}(i = 1, 2, \ldots, n)\) models n services, a tuple \(CoWFN = (oWFN_1, \ldots, oWFN_n, IT, FI)\) is called as Composition of open Workflow Nets by MTs if and only if:

1. \(IT\) is the set of MTs between oWFNs;
2. \(FI\) is the set of arcs between the \(IP_i/OP_i\) in \(oWFN_i(i = 1, 2, \ldots, n)\) and MTs.
Fig. 3. Two examples of complex conditions.

Reconsidering the previous scenario, by analyzing the interfaces of eShop and TPC services in Fig. 1, the MMs are obtained as follows:

\[
\langle \text{eShop. ServiceID, TPC. ServiceID}\rangle, \\
\langle \text{TPC(CardToken, ResServiceID, OrderID, UserID)}, \\
\text{eShop.(CardToken, ResServiceID)} + \text{eShop.(ResServiceID, OrderID, UserID)} \rangle, \\
\langle \text{eShop.(CardID, Password, OrderData)}, \\
\text{TPC.(CardID, Password, OrderData)} \rangle
\]

Based on the above MMs, we obtain three MPMs: \( \langle p_5, p_{14} \rangle \), \( \langle p_{15}, p_6 + p_8 \rangle \), and \( \langle p_r + p_0, p_{10} \rangle \). Then, FMT \( t_7 \), SMT \( t_8 \) and MMT \( t_6 \) are constructed, as shown in Fig. 2.

For the complex conditions that an output message place appears in more than one MPM, we cannot directly use the above basic MTs. The output message places for these MPMs need to be duplicated. This is do not affect analyzing of composition compatibility, because the corresponding tokens remain in the duplicated output message places are allowed for the compatibility of oWFNs (see Definition 9 in Section III).

By adding duplicated places, we assure that each output message place only exists in only one MPM so that complex conditions can be achieved by combining basic MTs.

For the example of the left of Fig. 3(a), the message of \( p_1 \) and a part of message \( p_2 \) are composed as the message \( p_3 \), and the rest of the part of message \( p_2 \) is used by the message of \( p_4 \). We get the MPMs \( \langle p_1 + p_2, p_3 \rangle \) and \( \langle p'_2, p_1 \rangle \). Here, a copy \( p'_2 \) of \( p_2 \) is inserted, and a MMT \( t_1 \) and FMT \( t_2 \) are constructed, as shown in the right of Fig. 3(a). Another example in the left of Fig. 3(b), its left MPM is divided into \( \langle p_{1} + p_{2}, p_{3}, p_{4} \rangle \) by duplicating \( p_1 \) and \( p_2 \). Subsequently, two MMTs \( t_1 \) and \( t_2 \) are constructed, as shown in the right of Fig. 3(b).

C. Mediation-Aided Composition of oWFNs

The procedure of mediation-aided composing oWFNs into a CoWFN is shown in the following Algorithm 1.

Different from mediators/adaptors in [7]–[15], MTs in our paper are high level and abstractive transitions, which hide unnecessary structural details irrelevant to the verification of composition. As conceptual mediators, MTs will be detailed implemented in the phase of automatically generating BPEL code of the composition.

Note that four kinds of basic control structures, namely, sequential, parallel, selective and iterative structures, have been defined in the Workflow Reference Model [7]–[10]. The iterative structure occurs when some transitions are executed iteratively. If oWFNs do not contain iterative structures, we can directly use Algorithm 1. Otherwise, we approximate the number of loops in a finite iterative structure and transform it to a sequence of transition by expanding cycles [23]. The input/output message places linked by transitions in an iterative control structure should also be duplicated into several “copied” places in the transformed model. As depicted in Fig. 4, a copy \( p'_4 \) of \( p_4 \) is inserted because the iterative structure is executed only one time.

Algorithm 1: Mediation-aided compose oWFNs

**Input:** oWFNs

**Output:** CoWFN

Step 1: According to WSDL specifications of exchanged messages among oWFNs, the MMs are set.

Step 2: For each MM \( \langle \text{source, target} \rangle \), a MPM is obtained by getting the place name of messages in \( \text{source} / \text{target} \), denoted by \( \langle p_{s_r}, p_{s_t} \rangle \).

Step 3: If each output message place of all oWFNs exist in only one MPM, then go to Step 5.

Step 4: Assume an output message place \( op \) appears in more than one MPMs, e.g., \( M_{p_{a}}, M_{p_{a+1}}, \ldots, M_{p_{b}} \), \( op \) is duplicated into \( \langle p'_{a_0}, op'_{a+1}, \ldots, op'_{a_b} \rangle \), and set \( \ast \text{op'}_{x}(a \leq x \leq b-1) \) is used to replace \( \text{op} \) in \( M_{p_x}(a \leq x \leq b-1) \).

Step 5: According to the MPMs, construct the MTs among oWFNs.
III. AUTOMATIC VERIFICATION OF COMPATIBILITY

Firstly, the concept of modular reachability graph (MRG) and its constructing procedure are presented in this section. Then, how to verify the compatibility of services by analyzing the MRG is discussed.

A. Modular Reachability Graph

First, a CoWFN is divided into a set of fragments (oWFNM)s.

Definition 6. (Open Workflow Net With MTs, oWFNM): Given oWFN = (P, T, F, IP, OP, F'), a 3-tuple oWFNM = (oWFN, C, F') is called as open Workflow Net with MTs if and only if:
1) C is the set of MTs; given ∀ c ∈ C, c = ∅ or c = ∅;
2) F' ⊆ (OP × C) \ (C × IP) is the set of interface arcs.

The CoWFN of eShop and TPC services in Fig. 2 can be divided into two oWFNM:s.

Intuitively, MRG is composed of one local reachability graph for each oWFNM, and a synchronization graph which captures their communications [17], [19].

Definition 7. (Modular Reachability Graph, MRG): Let oWFNM, (i = 1, 2, ..., n) be decomposed from a CoWFN. The modular reachability graph is a (n + 1)-tuple MRG = (RG1, ..., RGn, SG), where:
1) RGi = (V , Ei, fi), (i = 1, 2, ..., n), is local reachability graph of oWFNMi in which: (a) (V , Ei) is a directed graph, where V i = R(M0i), and Ei = {Mx | My ∈ R(M0i), ∃t ∈ Ti, M′x ≠ M′y}; (b) fi : Ei → Ti is a mapping from Ei to Ti in oWFNMi, fi(Mx, My) = t if Mx[t] > My.
2) SG = (V , E, f) is the synchronization graph among oWFNM:s, in which: (a) (V , E) is a directed graph, V ⊆ R(M01) × ⋯ × R(M0n); E = {(Mxv, Myv) | Mxv, Myv ∈ V, ∀ v ∈ IT; Mxv[t] > Myv}; (b) f : E → IT is a mapping from E to the set of IT, f(Mxv, Myv) = t if Mxv[t] > Myv.

A semiautomatically method of constructing MRG has been presented in [19], which needs the designers to manually decompose the composition into fragments. In this paper, we move one step forward to propose a full-automatically constructing and analyzing procedure from a CoWFN, which is shown in the following Algorithm 2.

First of all, we explain the operation of marking projection that will be used in the automatically constructing procedure.

Definition 8. (Marking Projection): The marking projection of M = (p1, p2, ..., pn) in SG on RGi is to remove the places from M that do not exist in the oWFN Mi, and is denoted with Γi(M).

For the CoWFN in Fig. 2, we construct the MRG = (RGeshop, RGtpc, SG) as shown in Fig. 5.

In order to prove of correctness of the Algorithm 2, we give the following Theorem 1.

Algorithm 2: Construct MRG from CoWFN

Input: CoWFN

Output: MRG

Step 1: For each MT t, we duplicate a set of copies, i.e., Ct = {t1, t2, ..., tk}, in which k is the number of oWFNs connecting.

Step 2: For i = 1 to n:

For ∀ p ∈ IPi of oWFNi, if p ∈ t, set p to be the input place of one transition t, of Ct. If p ∈ t, set p to be the output place of one transition t, of Ct. Then, delete t, from Ct.

Step 3: For each MT t, rename its copies t1, t2, ..., and tk to t.

Step 4: Construct each initial node Mi0 in the RGi(1 ≤ i ≤ n); then construct the initial node (M01 × ⋯ × Mi0) for SG.

Step 5: For i = 1 to n:

Parallel compute and draw the nodes for each RGi until no any enabled transitions in oWFNMi.

Step 6: Get the set of enabled MTs, denoted as ST, and for each t ∈ ST:

Assume t is enabled by M × ⋯ × Mj that Mi, ..., Mj are the markings of RG1, ..., RGj, the node (Mi × ⋯ × Mj) and arc <(M × ⋯ × Mj)[](M′ × ⋯ × M′j)> are construct in SG. Then, we construct the node of G1(M′ × ⋯ × M′j) in the RGi(1 ≤ i ≤ n), respectively.

Step 7: If there is not any possible enabled transition in oWFNM,s, the Algorithm ends. Otherwise, go to Step 5.

Theorem 1: Assume CoWFN = (oWFNi, ..., oWFNm, IT, FT) and its fragments denoted as oWFNMi(1 ≤ i ≤ n), the MRG of CoWFN is isomorphic with its traditional reachability graph.

Its proof is very similar with [17], and is omitted because of the limit of paper space.

B. Analyzing Based on MRG

Definition 9. (Composition Compatibility of oWFNs): The CoWFN of oWFNi(1 ≤ i ≤ n) are regarded as compatible, if its MRG satisfies the following cases:
1) For ∀ M ∈ RG1(1 ≤ i ≤ n), there exists a firing sequence σ and Mσ, satisfying Mσ > M and Mσ ≥ M (terminal marking).
2) Assume M ∈ RG, (1 ≤ i ≤ n) and M ≥ Mi, if ∃p ∈ P satisfying M(p) > M1(p), then p ∈ IP1 ∪ OP2 ∪ ⋯ ∪ OPn.

Before presenting the automatically analyzing procedure of compatibility by MRG, as shown in the following Algorithm 3, we explain the operation of cross-product of marking sets that will be used in it.

Definition 10. (Cross-Product of Marking Sets): Given two marking sets: S1 = {M11, ..., M1n} in RG1 and S2 = {M21, ..., M2n} in RG2. The cross-product of them is to combine all possible the markings from S1 and S2, and is denoted with S1 × S2.

For the example of MRG in Fig. 5, we conclude that eShop and TPC services are compatible by Algorithm 3.
TABLE I

<table>
<thead>
<tr>
<th>Specification of cases</th>
<th>Traditional approach</th>
<th>Our approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two oWFNs with 12 places, 8 transitions and 1 MT.</td>
<td>15 nodes, 29 arcs</td>
<td>11 nodes, 12 arcs</td>
</tr>
<tr>
<td>Three oWFNs with 20 places, 13 transitions and 2 MTs.</td>
<td>81 nodes, 231 arcs</td>
<td>22 nodes, 17 arcs</td>
</tr>
<tr>
<td>Four oWFNs with 29 places, 18 transitions and 3 MTs.</td>
<td>289 nodes, 2101 arcs</td>
<td>34 nodes, 25 arcs</td>
</tr>
<tr>
<td>Four oWFNs with 31 places, 20 transitions and 3 MTs.</td>
<td>387 nodes, 2300 arcs</td>
<td>35 nodes, 27 arcs</td>
</tr>
<tr>
<td>Five oWFNs with 40 places, 23 transitions and 4 MTs.</td>
<td>1497 nodes, 12281 arcs</td>
<td>49 nodes, 38 arcs</td>
</tr>
</tbody>
</table>

Algorithm 3: Analyze the compatibility by MRG

Input: MRG
Output: Results of compatibility

Step 1: For each \( M_i \in SG \), gets its \( \Gamma_i(M_i) \) on \( oWFN M_i (1 \leq i \leq n) \).

Step 2: For \( i = 1 \) to \( n \):
Take all possible projection markings on \( oWFN M_i \), and denote it as the set \( MS_i \).

Step 3: Construct the cross-product set denoted as \( CMS \) by \( \prod \{MS_1, MS_2, \ldots, MS_n\} \), and remove the projection markings from \( CMS \) which are labeled on \( SG \) arcs.

Step 4: For each \( M_o \in CMS \), gets its projection marking \( \Gamma_o(M_o) \) on \( oWFN M_o (1 \leq i \leq n) \) denoted as the set \( TMS_i \).

Step 5: If \( \forall M_i \in TMS_i (1 \leq i \leq n) \) satisfies the following cases, the composition is compatible:

1) \( M_i \in TMS_i \) is the logical state \( M_o \) of \( oWFN \); and
2) If \( \exists \psi \in P \) satisfying \( M_o(\psi) > M_e(\psi) \), then \( \in IP_1 \cup OP_2 \ldots IP_n \cup OP_n \).

Algorithm 4: Transform to BPEL code

Input: CoWFN
Output: The BPEL code of CoWFN

Step 1: Services are defined as the partner links by activity \(<parti Link>\). Then, the variables for composition are also defined by activity \(<variable>\) according to the messages of the WSDL description of services.

Step 2: According to the kinds of MTs, we construct the BPEL code for them in the format of ECA rule. Then, the code of all MTs are embraced by \(<event Handler>\) and \(<event Handler>\).

Step 3: For the some copied output places in the CoWFN, complete events are defined according to needs by activity \(<invoke>\), which do not carry any data.

Step 4: If there exist a \( MPM \) in the procedure of composition, we supplement the corresponding messages (parts or elements) between the pair of \(<assign>\) and \(<copy>\) to perform the transferring or transforming of messages.

Step 5: For each service, we use activity \(<while>\) to check whether or not the ECA rules (the code of MTs) have been triggered.
true, to perform the interface operations by activities \(<\text{receive}>\), \(<\text{invoke}>\) or \(<\text{reply}>\).

Step 6: The BPEL code blocks of each service is embraced by \(<\text{process}>\) and \(</\text{process}>\), in order to make services execute concurrently.

Step 7: To specify the CoWFN receiving the initial message that starts it, the initial activity \(<\text{receive}>\) is added with \(<\text{createInstance}>\) attribute. Then, the BPEL code of CoWFN is obtained by putting all the above code blocks into \(<\text{process}>\) and \(</\text{process}>\).

Note that the generated BPEL code lack specific implementation details which cannot be automatically derived from a CoWFN by the above Algorithm 4, so the code needs further refinement.

V. PROTOTYPE SYSTEM

In this section, a prototype system (Mediation-aided Composition System of Services, MCSS) implementing the proposed approach is presented.

As shown in Fig. 6, MCSS has five components: Petri net translator, adding MTs modeler, MRG constructor, verification analyzer, and BPEL code generator. Petri net translator adopts the free software BPEL2oWFN [21] and the other four modules are developed based on the open-source software PIPE [22]. The original PIPE uses the Model-Controller-View architecture pattern to implement several Petri net analysis plug-in modules. Based on the existing architecture of PIPE, the modules in MCSS (adding MTs modeler, MRG constructor, verification analyzer, and BPEL code generator) can be developed quickly.

Petri net translator is responsible for translating BPEL services into oWFNs in the file format of Petri Net Markup Language (PNML). We adopt the existing free software tool BPEL2oWFN. The output PNML files of oWFNs are input into adding MTs modeler. Then, adding MTs modeler composes the oWFNs into a CoWFN. It merges the PNML files of oWFNs as the whole file of a CoWFN by adding XML sections of MTs. The designers can manually add MTs based on the WSDL descriptions of oWFNs.

The whole PNML file of CoWFN is the input of MRG constructor. MRG constructor serves as the engine to automatically decompose a CoWFN into some oWFNs in the background through manipulating its PNML file. The PNML files of oWFNs are put into the fragment pool and are used to support verifying the compatibility of composition based on MRG. In order to intuitively illustrate the MRG, MCSS return the graphic results to the designers.

Based on the above MRG, verification analyzer determines whether the composition is compatible. If the compatibility is validated, BPEL code generator is enabled for the designers to get the BPEL code of composed service. Otherwise, the error messages are returned.

BPEL code generator can automatically generate the BPEL code from CoWFN in the format of ECA rules. All code are implemented and stored in the BPEL template library. Here, the library includes the predefined BPEL template for three kinds of MTs, and message variables for composition according to their message formats.

In order to illustrate the executing procedure of MCSS, we adopt the example of the eShop and TPC services described in Section II. As shown in the Fig. 7, the oWFNs of eShop and TPC services are output by Petri net translator. Based on adding MTs modeler, the designers can add MTs \(t_1\), \(t_2\), and \(t_3\), and obtain their CoWFN.

Next, MRG constructor automatically generates the MRG in the background and demonstrates the MRG in the graphic format as shown in Fig. 8, where the node color of RGs for eShop and TPC is light blue, while the node color of SG is dark red.

Afterwards, verification analyzer returns the result that the composition is compatible and the BPEL code generator is enabled.
the whole BPEL code of the example from the CoWFN is automatically generated by the BPEL code generator.

Because of the limit of paper length, the screenshots of other components and the whole BPEL process of example are not shown in this paper.

VI. RELATED WORK

There are some related studies [7]–[15] on the issue of compatibility verification and automatic generation of BPEL code in mediation-aided composition.

A. Compatibility Verification

Mooij and Voorhoeve [7] address the automated generation of adapters based on the open Workflow Net (same with our work in this paper). Then, correctness of adapters is verified by proving the properties (e.g., deadlock-freedom) holding for the composition of services. Wang [8] proposes a visual language for specifying adapters for services and use Petri nets to check their correctness. Tan et al. [9] transform the BPEL services into service workflow nets (a special class of Petri nets) and analyze the compatibility of two services based on mediators. Based on Colored Petri Nets (CPNs), Li et al. [10] present a heuristic approach to identify the protocol mismatches of services and select appropriate mediator patterns. Then, if the composition is verified, the BPEL templates of mediator patterns are also developed. Guermouche et al. [11] model services as automata models. Then, the mediator are generated to supply the missing messages which are required to complete the Cartesian product of automates. Simultaneously, the correctness of constructing adapters is verified. Also, based on automata, Nezhad et al. [12] present a method for identification of the split/merge class of interface mismatches and a semiautomated matching approach to construct an adaptor for these services. Bachir and Fauvet [13] check whether two services based on Finite State Machines (FSMs) are incompatible, and provides the locations in the service interfaces where these incompatibilities occur. So that mediators can be constructed subsequently. Zhou [14] obtains abstract protocols from service protocols by a set of rules. Then, they construct adaptation matrix, using an adapted depth-first search with back tracking technique, so that the conditions that these two services can be adapted is identified. Canal et al. [15] present an approach to automatically generate adaptor based on Labeled Transition Systems (LTSSs). Along with adapters is constructed, their correctness is also verified.

The above methods [7]–[15] can address the issue of compatibility verification for a small number of services. However, they all suffer from the problem of state space explosion when verifying the composition of complex services, especially the number of services is large.

B. Automatic Generation of BPEL Code

The abovementioned methods [7]–[9], [11]–[15] only propose conceptual mediators, e.g., the function of mediators such as how to transfer or split messages, and cannot produce the (abstract) BPEL code from composition models to support execution. Li et al. [10] present the BPEL templates of several mediator patterns. However, they all do not consider how to automatically generate the whole BPEL code from the composition models.

VII. CONCLUSION

The mediation-aided composition is attracting more attention when dealing with incompatibilities of services. But existing approaches suffer from the problem of state space explosion and cannot automatically generate the BPEL code. The approach presented in this paper has addressed the weaknesses of prior ones. Specifically, the compatibility of composition is verified by automatically constructing and analyzing its MRG, which can alleviate the state space explosion problem. If the composition is verified to be valid, the BPEL code is automatically generated, which can significantly save the time and labor of designers with the fast changing need.

In the future, we plan to extend our approach in two aspects.

1) The present approach needs designers to give MMIs based on service WSDL specifications. We will explore how to adopt semantic technology to automatically generate MMIs.

2) In a practical business environment, some service composition often has timed specification (or temporal constraints) [23]. We will extend MRG to express the timed state space.

REFERENCES


