

Transforming a formal business process model into a service interaction model

Zonghua Li¹, Zhengwei Ye²

¹ School of Computer Science and Technology, Huaiyin Normal University, 223300 Huaian, Jiangsu, China

² School of Urban and Environmental Sciences, Huaiyin Normal University, 223300 Huaian, Jiangsu, China

Correspondence author: Zonghua Li(leeleaf@163.com)

(Received April 28, 2021, accepted September 12, 2021)

Abstract: The business process model and service interaction model represent the behaviour model on business view and system view respectively. The automatic transformation from the business process model to the service interaction model can effectively narrow the gap between the business view and the system view. Thus, in this paper, a model transformation approach that can transform the business process model described by extended Petri net into service interaction model in system view is proposed. This approach by extending the UML sequence diagram meta model represents service interaction model, and by using model-driven development designs the transformation from extended Petri net to service interaction model. Especially, each situation of the behaviour transition linked to the different types of places in extended Petri net model are detailed analyzed. According with the these situation, the transformation rules from each situation to different types of message in the service interaction model are designed. In order to carry out the automatic executing of the formalization transformation, the transformation rules are described by the ATL model transformation language, and the ATL codes are implemented by Eclipse platform. Finally, the executing result of the model transformation plug is demonstrated by the Travel Agency system. The experimental results show that the approach in this paper can realize the automatic transformation from business process model to service interaction model, and improve the efficiency of software development.

Keywords: business process model; extended Petri net; service interaction model; model transformation

1. Introduction

The business process model can describe the service process of the business system. Modeling the business process is one key stages in the development process of the business system. There are many ways to describe the business process, such as the business process modelling notation (BPMN) [1], Activity diagram model [2], Petri net model [3], etc. The business process model is the computation independent model (CIM) level model in the model driven architecture (MDA). Currently, many researches are based on model transformation to map business process model into related models at the PIM level, such as bpmn2usecase [4-5], valuemodel2usecase[6], bpmn2statediragram [5] and bpmn2classmodel [7], etc. However, these transformations are difficult to verify the correctness and completeness of the model without formalization. It is well known that the formal process modeling can solve problems such as deadlocks and conflicts of business process models [8]. At the same time, in order to verify the correctness, completeness, and effectiveness of business process models, the formal description of business process model has become an important way of model verification. Among many formal methods, Petri net, as a general formalization method, use "flow" to analyze the behavior of the system [9]. Furthermore, in order to describe more process details, many researchers extend Petri net to study more process details from exception handling [10], context awareness [3], and complex time processes [11], etc.

The service interaction model (SIM) is based on the UML sequence diagram model. Using sequence diagram grammar and symbols, SIM model describes the execution sequence of a series of activities between service objects in order to complete a certain business service, and describes the time order of the message transmission between service objects. The execution sequence of activities in the SIM model is consistent with the business execution process in the business process model, and the message transfer between the service objects is also related to the object flow in the business process. However, there are few researches focused on the transformation from business process model into service interaction model in the current

researches. Therefore, an approach that transforms formal business process model into a service interaction model from is proposed in this paper.

This work makes the following main contributions: (a) focused on a business process perspective, we analyze the extend Petri nets model (EPN) elements and SIM model elements, and discriminate different business flow details; (b) the transformation rules from EPN model to SIM model are designed, and the automatic transformation have been implemented in the Eclipse framework by integrating ATL language.

The remainder of this paper is organized as follows: Section 2 analyses the main related work that involves the transformation of the formal business process model. Section 3 describes the architecture of the EPN model to SIM model. The metamodel definitions of the EPN model and the SIM model are described in Section 4. Section 5 demonstrates the mapping using a case study. Finally, we discuss the primary contributions, conclusions, and future work in Section 6.

2. Related Work

The Petri net model can formally describe control flow, object flow and information flow effectively, and has complete semantic description. Therefore, many researches use Petri net to formalize the BPMN model and UML activity diagram. For the BPMN formal transformation, some researches [8,12-13] use object Petri net model and colored Petri net model to describe the BPMN model elements. While Dijkman et al.[14-15] focused on the semantic analysis, transform the BPMN model to the Petri net model, and verify the effectiveness of the business process. In addition, using Petri nets can also perform reliability verification [16-17], attribute verification [18], security verification [19] and semantic specification [20] for BPMN models. For the UML activity diagram formal transformation, Trickvoic [21] proposes a complete process of mapping UML activity diagram into Petri net model to verify the dynamic model of the real business system. However, since the Condition Event Net (CEN), as a basic Petri net, cannot accurately model workflows, Eshuis et al. [22] discusses the execution semantic design choices of the Petri net model and UML activity diagram. In view of the importance of the object dynamic semantics, Bouabana-Tebibel et al. [23] use object Petri nets to verify the correctness of object dynamic semantics in UML activity diagram. In order to simplify the formalization of the UML activities, Staines [24] proposed a solution, which transform UML activity diagram into colored Petri nets, so that it has a graphical visual verification function. For the Petri net modelling tools, such as ePNK, PNML Framework, Coloane, Tina, etc., can allow intuitive and visual modelling Petri net, so it is easy for system analysts and software developers to understand. Thus, the Petri net model has become the most common formal tool for business process model.

SIM model is mainly used for interactive modeling in the system view, and the execution results can be analyzed using Petri net [25-26]. For the formal research of UML sequence diagrams, Faria et al. [27] propose a method of formalizing UML sequence diagrams using colored Petri nets. They define the mapping rules of UML sequence diagrams to colored Petri net models, and analyze the execution results of this colored Petri net. For the real-time and embedded (MARTE) systems, Yang et al. [26] formalize the MARTE-type timing diagram as a time-colored Petri net model with suppressed arcs. The advantages of Yang's method are the concentration on verification of the time property.

Although the current researches formalize the business process model and service interaction model respectively to verify their correctness and effectiveness, there is still a large gap between the business analysis stage and the system modeling stage. In order to narrow the gap, some researches have proposed transformation methods from business view model to system view model based on MDA (Model Driven Architecture) technology. These transformations include from BPMN model to use case model [5], BPMN model to class model [4], value model to use case model [21], BPMN model to service composition process [28], etc. However, since the SIM model needs to design the message details between business entities, there is few researches on automatically transforming the business process model on the business view into the SIM model on the system view. Although the UML sequence model in the system view is mentioned by Bousetta et al. [4], the sequence model is completely manually designed and refined by software designers on the basis of use case model.

In order to transform the formalized business process model to the SIM model, in this paper, we propose a approach that can support the automatic transformation from formal business process models to SIM model, compared with previous studies, our approach focused on the automatic generation of SIM model, design the transformation rules to realized the automatic transformation from formal business process model to SIM

model by using Eclipse framework. The main advantage of our approach is to reduce the workload of the software design, and to narrow the gap between business view and system view.

3. Overview of model transformation architecture

A MDA development process focusing on formal model is proposed as shown in Figure 1. In this development process, we use extended Petri net model to describe business process requirement, and use service interaction model to describe the interaction behaviour of the information system. The service interaction model using UML sequence notation is the target model which is mapped by extended Petri nets model, and the extended Petri nets notation can be implemented by ePNK tool in Eclipse framework. In this approach, since the business process model is depicted by formal model, the correctness and the completeness of the business process model can be verified directly by extended Petri net. Thus, mapping the extended Petri net model into the service interaction model directly can reduce the gap between the business view and the system view model.

Figure 1 shows the model transformation process that is based on formalised way, the highlight of our approach is that it can avoid the formalization again for the service interaction model. Furthermore, we design the transformation plug-in (EPN2SIM) to implement the automatic transformation. The transformations component can map the source model into target model using ATLAS Transformation Language (ATL). This approach is different from other researcher's model transformation approaches because our approach transfer the formal model into service interaction model. This transformation can generate the SIM model automatically, and narrow the gap between the business view and system view.

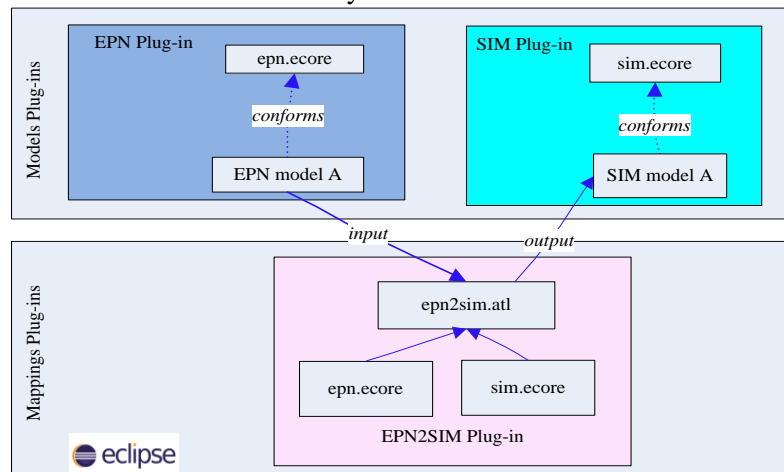


Figure 1: Architecture of the transformation process of the EPN model to SIM model

4. Model transformation process

4.1 Metamodel definition

(1) Extended Petri nets model

The EPN model adds elements that describe the static organization structure and dynamic behavior of the business system on the basis of the conditional event Petri net. Based on the previous research [29], the metamodel of the extended Petri net model used in this paper is shown in Figure 2. It can be seen that the metamodel elements of the EPN model are composed of *InnerPlace* (represented by a circle), *OuterPlace* (represented by a dotted circle), *BehaviourTransition* (represented by a rectangle), *SilentTransition* (represented by a solid rectangle), *SubPetriNets* (represented by a rounded rectangle) and *OrganizationIdentifier*. Among them, *OrganizationIdentifier* (OI) and *GroupIdentifier* (GI) represent external and internal actors of the business system respectively; The place is refined into *InnerPlace* and *OuterPlace*, representing the execution order of business object actions and the interaction cooperation between different actors respectively. *InnerPlace* is used to represent the control flow information between different object nodes, and *OuterPlace* is used to represent the message flow interaction information between different actors in the business system. Therefore, *InnerPlace* cannot cross the boundary of OI entities, and OI entities are linked through *OuterPlace*. The transition element is refined into *BehaviourTransition* and *SilentTransition*, where *SilentTransition* represents

static events, which is used to represent the beginning and ending of business processes and capture the path information of business control flow and message flow; *BehaviourTransition* represents dynamic events, and is used to represent the behavior in the business process.

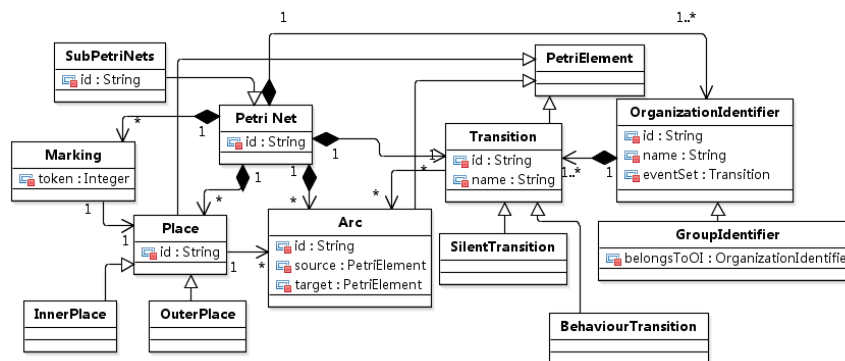


Figure 2: Extended Petri Net metamodel (modified from [29])

(2) Service Interaction model

Interactions usually are used in many different situations. They can gain a better grip of an interaction scenarios for an individual designer or for a team that needs to achieve a common understanding of the situation [2]. Interaction is a series of service behavior, which focuses on message exchange between the service entities. SIM describes the execution sequence of sets of activities between service objects in order to complete a business service. Figure 3 shows the meta model of the SIM, which includes *Interaction*, *ServiceEntity*, *Gateway*, *Message*, *LifeLine*, *OccurrenceSpecification*, and so on. Particularly, *LifeLine* denotes a service entity playing a specific role in a service interaction process; *Message* represents the interaction information between service entities in a service interaction process; *Gateway* denotes the control logic of the business task under some conditions; *Interaction* is a unit of service behavior, focusing on information exchange between connectable entities. Since UML2.0 is a standardized modeling language familiar to software analysts and software developers, this paper uses UML2.0 sequence diagrams to represent the SIM model, and adds service-related *ServiceInteraction* and *ServiceEntity* elements on the basis of the UML sequence diagram meta-model. Among them, *ServiceInteraction* represents service interaction, which represents the details of information exchange between multiple service entities; *ServiceEntity* represents a service entity, which is a derived element of the *lifeline*, and represents the entity element of a business service. The model is transformed from the EPN model on the business view, and more service interaction details are added in the model refinement process.

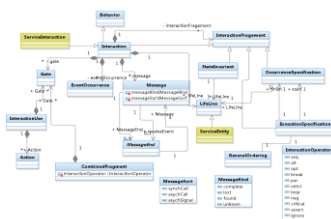


Figure 3: Partial view of SIM metamodel (modified from [2])

4.2. Model Transformation

Based on the metamodel definition of the SIM, the *lifeline* represents the existence of a *ServiceEntity*. The *ServiceInteraction* represents an interaction between user and business system, which reflects the message delivering in two service entities. *Message* defines a special form of the communication. It not only points out the execution description of the communication form, but also specifies the sender and receiver for a message. Therefore, the message plays an important role in a SIM model. So, we define the mapping rule between EPN and SIM using natural language (listed in Table 1) and graphical markers (listed in Table 2 and Table 3). Table 2 shows the details of how the *BehaviorTransition* node is transformed to various message nodes in the SIM model in the case of multiple input arcs and output arcs. Table 3 shows that in addition to the silent transition that represents the beginning and the end, the other *SilentTransition* that represents the execution path selection is mapped to the mapping rule of the combined fragment (*Combinedfragment*) element in the SIM model.

Table 1. Mapping rules and mapping rule action from EPN to SIM

| Mapping type | Source element (EPN) | Target element(SIM) | Mapping rules |
|-----------------------------------|-------------------------------------------------|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Behavior transition-based mapping | Behaviour transition+one inner place | Message | A behavior transition node has only an input arc and an output arc, and the pre-condition and post-condition are inner place. This behavior transition is mapped onto a self-message in SIM model. |
| | Behaviour transition+outer place | Message | A behavior transition node has an input arc and an outer place. This behavior transition node is mapped onto a synchronization call message. The sender of the message is the OI to which the behavior transition belongs, and the receiver of the message is the OI to which the post condition linked behavior transition belongs. |
| | Behaviour transition+more than one inner place | CombinedFragment(loop) | The input arc and the output arc linked a behavior transition node is the same inner place. This behavior transition node is mapped onto a loop segment in SIM model. |
| | Subpage | InteractionUse | A SubPetriNets node in EPN model can be mapped onto a InteractionUse in SIM model |
| Silent transition-based mapping | Silent transition+marking | Lifeline | A silent transition linked an initial place that is marked with a token and without an input arc. In this case, the silent transition is mapped onto a lifeline element in a SIM model, and the name of the lifeline is mapped by the name of OI element which the silent transition belongs in the EPN model. |
| | Two silent transitions+an inner place | CombinedFragment(alt) | An inner place node linked two silent transition by output arc. This situation may be mapped onto an alt segment in SIM, Each of the silent transition nodes are mapped the conditions in the combination fragment. |
| | One silent transition+an inner place | CombinedFragment(par) | A silent transition node has many parallel output arcs, linked these output arcs may be the inner place nodes, or the outer place nodes. This situation may be mapped onto the fork node in a par combination segment. |
| | One silent transition+more than one inner place | CombinedFragment(par) | A silent transition node has many parallel input arcs, linked these output arcs may be the inner place nodes, or the outer place nodes. This situation may be mapped onto the join node in a par combination segment. |
| | Two silent transitions+two inner places | CombinedFragment(opt) | An inner place node linked silent transition node by two input arcs. This situation may be mapped onto the opt combination segment. Each silent transition node represent the opt options. |

Table 2. Behavior transition-based mapping

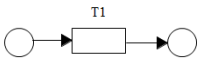
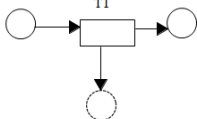
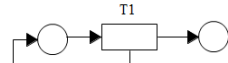




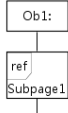
| Operation element | Behaviour transition+one inner place | Behaviour transition+outer place | Behaviour transition+more than one inner place | Subpage |
|---------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| Source module (EPN) |  |  |  |  |
| Target module (SIM) |  |  |  |  |

Table 3. Silent transition-based mapping

| Operation element | Silent transition+marking | Two silent transitions+an inner place | One silent transition+an inner place | One silent transition+more than one inner place | Two silent transitions+two inner places |
|---------------------|---------------------------|---------------------------------------|--------------------------------------|-------------------------------------------------|-----------------------------------------|
| Source module (EPN) | | | | | |
| Target module (SIM) | | | | | |

The transformation algorithm from behavior transition node in EPN model to message element in SIM model is presented by pseudocode in Algorithm 1. The algorithm takes a *epn* instance file as input and outputs a *sim* instance file for the EPN model. The design details of the transformation are as follows:

- If the input arc and the output arc of the behavior transition node in EPN model are connected by inner place, it needs to be transformed into self-message in SIM model, that is, the sender and receiver of the message are OI entities of the behavior transition node.
- If the input arc of the behavior transition node in EPN model is connected to the outer place, it indicates that the transformed message in SIM model should be the interaction between the two different service entities. So, the OI entity of this behavior transition node is the receiver of the message.
- If the output arc of the behavior transition node in EPN model is connected to an outer place, it indicates that the transformed message in SIM model should also be an interaction between two different service entities. So, the OI entity of this behavior transition node is the sender of the message.
- If the input arc and the output arc of the behavior transition node are connected to the same inner place, it indicates that the mapped message should be a Loop segment.

However, the types of messages in SIM model include synchronous call (*synchCall*), asynchronous call (*asynchCall*) and asynchronous signal (*asynchSignal*). Therefore, the transformed message needs to be adjusted manually by the system analyst.

Algorithm 1: Transformation Algorithm from behavior transition node in the EPN model to message node in the SIM model.

```

Input: An EPN model epninstance.pnml
Output: A SIM model siminstance.sqd
1 Collections:
2    $bt_i.T = \{bt_1, bt_2, bt_3, \dots\}$  // object list of the behavior transition nodes declared in the epninstance.pnml file.
3    $a_i.A = \{a_1, a_2, a_3, \dots\}$  // object list of the arc nodes declared in epninstance.pnml file.
4 for each Behavior Transition  $bt_i$  in  $bt_i.T$  do
5   int  $i=0, j=0$ ;
6   Organization Identifier  $oi = bt_i.belongtoOI()$ ;
7   Create a set tset to save the arcs which the target node is oi ;
8   Create a set sset to save the arcs which the source node is oi ;
9   for each Arc  $a_i$  in  $a_i.A$  do
10  //If the target node of an arc in the EPN model is behavior transition node, save the arc to the tset .
11  //If the source node of an arc in the EPN model is behavior transition node, save the arc to the sset .
12    if  $a_i.target == bt_i$  then
13       $tset = tset \cup \{a_i\}$  ;
14       $i=i+1$ ;
15    end if
16    if  $a_i.source == bt_i$  then
17       $sset = sset \cup \{a_i\}$ 
18       $j=j+1$ ;
19    end if

```

```

20  end for
21  if  $i \geq 1$  &&  $j = 1$  then
22    Create an Execution Specification  $es_i$  in  $oi$ ;
23  Create a message  $m_i$  in  $oi$ ;
24     $m_i.name = bt_i.name$ ;
25     $m_i.receive = es_i$ ;
26  else
27  for each Arc  $a_i$  in  $sset$  do
28    if  $a_i.target.type == "OuterPlace"$  then
29      //Query the next behavior transition node  $bt_i$ 
30      Query Organization Identifier  $target\_oi$ ;
31      Create a message  $m_i$  in  $oi$ ;
32         $m_i.name = bt_i.name$ ;
33         $m_i.send = o_i$ ;
34         $m_i.receive = target\_oi$ ;
35    end if
36  end for
37  for each Arc  $b_i$  in  $tset$  do
38    if  $b_i.source.type == "OuterPlace"$  then
39      //Query the previous behavior transition node  $bt_i$ 
40      Query Organization Identifier  $source\_oi$ ;
41      Create a message  $m_i$  in  $source\_oi$ ;
42         $m_i.name = bt_i.name$ ;
43         $m_i.send = source\_oi$ ;
44         $m_i.receive = o_i$ ;
45    end if
46  end for
47  for each Arc  $b_i$  in  $tset$  do
48    for each Arc  $c_i$  in  $sset$  do
49      if  $b_i.source.type == "InnerPlace"$  and  $c_i.target.type == "InnerPlace"$  then
50        if  $b_i.source == c_i.target$  then
51          Query Organization Identifier  $oi$ ;
52          Create a Combined Fragment  $cf$  and interactionOperator=loop;
53             $cf.name = bt_i.name$ ;
54        end if
55      end if
56    end for
57  end for
58  end if
59  end for

```

Significantly, the interaction operator of the *Combined Fragments* element in SIM model is composed of *alt*, *opt*, *loop*, *par* and etc. An *alt* interaction operator represents the choice of a behavior, There are multiple operands that could be chosen in the ATL fragment. But, the optional operands must have the explicit or implicit conditional expression, and the result of the conditional expression is true at a certain point of interaction. Therefore, if there is an inner place node in the EPN model, the output arcs of this inner place are connected by silent transition nodes, in this case, the silent transition nodes in EPN model should be transformed into *alt* interaction operator in the SIM model, where each silent transition branch of the inner place is a conditional expression in *alt* fragment. The *opt* fragment is used to represent a decision node and it also represents a merge node correspondingly. Therefore, if there is an inner place and the source nodes of its multiple input arcs are silent transition nodes in the EPN model, the silent transition nodes should be transformed into *opt* composite fragments in SIM model. The *par* fragment is used to represent a fork node and it also represents a join node correspondingly. Thus, if there is a silent transition node in EPN model, which has multiple balanced input arcs or output arcs, the silent transition node should be transformed into *par*

fragment in SIM model. The *loop* fragment represents a simple loop. So, if there is an inner place node, the input arc and the output arc linked the same behavior transition node, then the behavior transition node in the EPN model is transformed into a *loop* combination fragment in the SIM model.

The *InteractionUse* element is a interaction in SIM model. it is represented as a special interaction fragment and an interaction is often called by reference way. Therefore, it is necessary to transform the subpage element in the EPN model into the interaction use in the SIM model. Especially, when the connection between two EPN models is mapped into two *InteractionUse* elements of SIM model, an ATL operation (rule addasynch) is designed in the transformation plug-in, which automatically adds synchronous call messages (synchCall) between the two *InteractionUse* elements to realize the call operation between the two *InteractionUse* elements.

4.3. Prototype development

The Eclipse Modelling Framework (EMF) [30] can define, edit and process a meta-model file (.ecore). Figure 4 shows the use of EMF to define the EPN and SIM meta-models on the Eclipse platform. The Petri Net Markup Language (PNML) [31] is an XML-based exchange format that supports graphical modeling. Many current Petri net modeling and analysis tools (such as ePNK, PNML Framework, Coloane, etc.) support the reading and editing of PNML files. Thus, the EPN model in our proposal is described in PNML language, and its model editing and processing are implemented by ePNK plug-in that integrates with the Eclipse platform. And the editing and processing of the SIM model is based on the UML2.0 sequence diagram model.

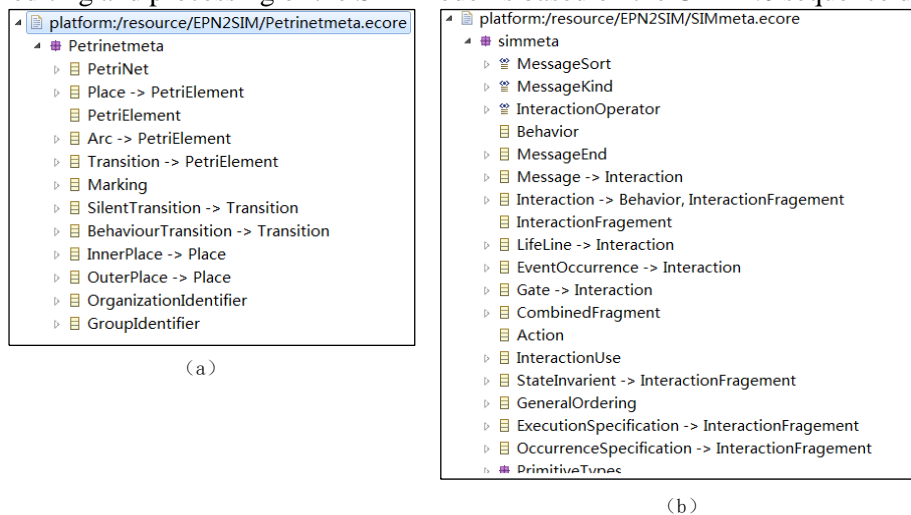


Figure 4: Metamodel definition: (a) EPN metamodel definition, (b) SIM metamodel definition

According to the transformation rules listed in Table 1, our approach integrates the ATL language to perform the transformation of the EPN2SIM on the Eclipse platform. The EPN2SIM transformation plug-in takes the EPN model as the source model and the SIM as the target model. The detailed transformation process is as follows: first, we use the EMF plug-in to create the EPN metamodel and the SIM metamodel; then, we describe the transformation specification according to the ATL syntax; finally, the model execution engine is used to implement the transformation from the source model to the target model.

In order to illustrate the automatic execution process of the EPN2SIM plug-in, Figure 5 shows the partial view of the model transformation from the behavior transition node to the message node based on the Algorithm 1. There are three transformation situations under the different conditions defined in epn2sim plug-in, they are behavior transition to *self message* element, behavior transition to *asychSignal* element, and behavior transition to *loop* fragment element. Take the behaviortransition2selfmessage transformation as an example, if a behavior transition node has only one input arc and at least one output arc (lines 139-140), and the input arc and output arc are connected by inner place (line 141), then this behavior transition node will be transformed to the *self message* in the SIM model, and its id and name values are directly transformed to the id and name of the self message element. Therefore, the sender and receiver of the self message element are both the OI which the behavior transition node belongs (lines 144-148).

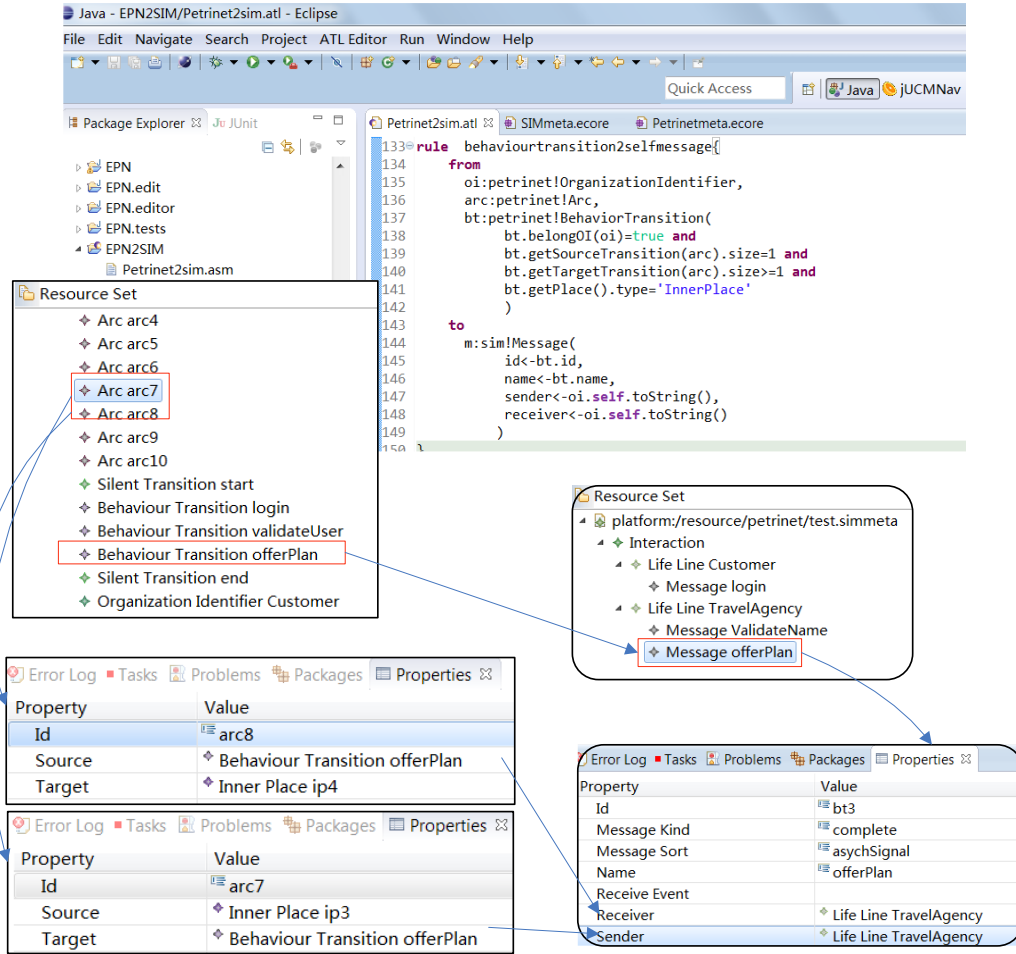


Figure 5: Partial view of EPN model mapping to SIM model and behaviourtransition2selfmessage ATL code excerpt

5. Case study

5.1. Scenario

The following scenario is provided by the World Wide Web Consortium (W3C) organization [32], in the following, we depict the scenarios of the Travel Agency System. Firstly, the customer submits a plan to the travel agency, including travel destination, departure time and return time, hotel standards and vehicles. Next, the travel agency evaluates to whom a Broker Agent can offer the service based on the customer criteria and link it. Each broker agent may offer a travel plan to the travel agency according to customer’s criteria. Then, the travel agency provides customers with a set of travel options that meet the conditions. Once the customer selects one of them, the customer needs to provide credit card information to the travel agency, and the travel agency will deal with the payment together with the corresponding finance company. After checking whether the payment is correct, the travel agency asks the corresponding broker to confirm the booking(s) and then to notifies the customer.

5.2. Models

In this paper, we use our proposed GSP modelling approach [29] to model the formalized business process model of the Travel Agency. The formalized business process model of this system is shown in Figure 6. Based on the transformation definitions from EPN to SIM in Section 4.2, taking the EPN model shown in Figure 6 as the input model, execute the EPN2SIM plug-in to obtain the SIM model (output model) as shown in Figure 7. It can be seen from Figure 7 that all behaviour transition nodes in the EPN model are transformed to message operations in the SIM model. The four inner places with tokens are transformed into four lifeline elements in the SIM model since these inner places are linked to the silent transition node. Significantly, the name of the lifeline element is determined according to the OI element to which the silent transition belongs. But the sub Petri net is transformed to the interaction use element in the SIM model.

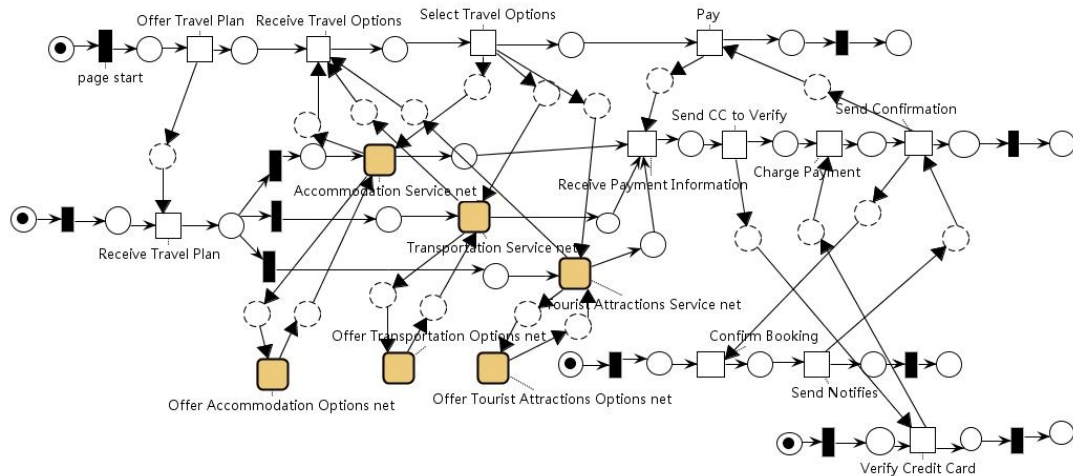


Figure 6: The EPN model of the Travel Agency [29]

In Figure 6, the customer needs to interact with the travel agency 4 times to obtain travel services in the interaction process. In these four interactions, the Offer Travel Plan, Select Travel Options and Pay represent the operation of the customer actively sending messages to the travel agency, but the Receive Travel Options represents the customer receiving the message from the travel agency. These four interactive operations are consistent with the several operations that the customer needs to complete in the SIM model. Therefore, the use of the EPN formal model can realize the automatic transformation of the business process model in business view to the behaviour model in system view.

However, the three sub Petri net models, such as *Accommodation Service*, *Transportation Service*, and *Tourist Attractions Service* in Figure 5, have not been further refined, thus, the *InteractionUse* element in transformed SIM model shown in Figure 7 need to refine and improve manually by business analysts. So, the refinement process of the SIM model is: (1) identify the entities or objects involved in the initial SIM model as lifeline elements in the SIM model. In the Travel Agency system, the travel plan requirement (*Customer'sNeed*) proposed by the customer and the hotel entity (*Accommodation*) queried by the travel agency should be extracted as lifeline elements in the SIM model; (2) refine the message operations of the SIM model; (3) add the necessary return message operation in the SIM model. For example, in the *Accommodation Service*, in order to realize the accommodation service, *TravelAgencyClientForm* object interacts with *Customer'sNeed* object and *BrokerAgent* object many times, and the *BrokerAgent* object directly interacts with the *Accommodation* object. Therefore, a return message operation should be set for each interaction. The service interaction model of the refined *Accommodation Service* is shown in Figure 8.

5.3. Comparison to other approaches

Compared with the majority of previous proposals, although some transformation approaches can execute automatic or semi-automatic mapping from business view to system view, these approaches lack well-defined formal semantic to maintain the consistency between source model and target model. In contrast, our approach transform the business process model after formal verification into the service interaction model in the system view directly. It is the advantage of our proposal which can be of help for the software designers that can refine and improve the model on the basis of this transformed model. At the same time, the interactive entity objects, interactive fragments and interactive messages in SIM model are more consistency with business process model by formal model transformation. Thus, our approach can narrow the gap between the business view and system view, and better guide the design and development of software.

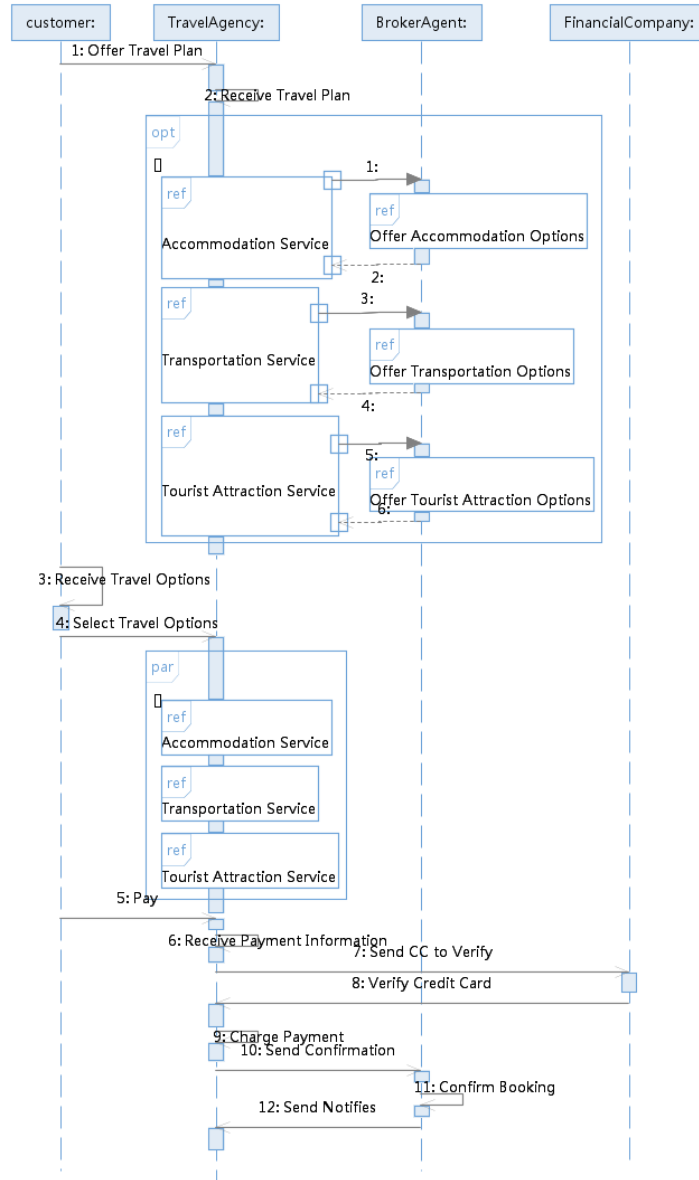


Figure 7: The SIM model transformed by EPN model

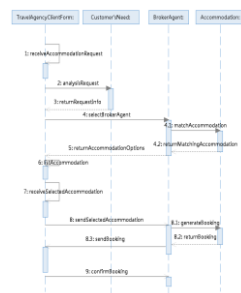


Figure 8: The SIM model of the Accommodation Service

6. Conclusions

In order to narrow the gap between the business view model and system view model, in this paper, we propose the automatic transformation from formal business process model to service interaction model. In particular, we have analyzed the details of the different interactions among the behavior transition, inner place and outer place elements in the EPN model. Furthermore, we have defined the conditions of each interaction situation, and designed the transformation rules of these interaction situation to different message elements in the SIM model. These transformations are realized based on EMF plug-in and ATL model language. The case study shows that the inner place and outer place can be effectively transformed into different message elements, and different gateways can be transformed into combined fragment elements. Thus, these transformation can transform the details of business process into the content of service interaction. Therefore, our approach can automatic transform the verified business process model into SIM model using the developed model transformation plug-in we designed. Overall, on the one hand, our proposal make that the software designers do not need to rebuild the SIM model from the beginning, and can improve the consistency between the business view model and the system view model. On the other hand, our proposal can also reduce the development cost.

However, in order to ensure the execution of the model transformation, we set some restriction conditions in EPN2SIM plug-in, e.g., the message type is assumed to be "complete", and the target model element occurrence specification and event occurrence need to be manually identified by software analysts, and the detailed interaction require manual refinement and improvement by software designers. Therefore, how to transform and generate the SIM model more automatically needs to be solved in our next work.

Acknowledgement

This research is supported by the National Natural Science Foundation of China (61902141); the MOE (Ministry of Education in China) Youth Foundation Project of Humanities and Social Sciences(19YJCZH095); and the Natural Science Foundation of Jiangsu Education Department of China (18KJB520006).

References:

1. OMG, Business Process Modeling Notation (BPMN) Version 1.0, OMG Final Adopted Specification, Object Management Group, 2006
2. Unified Modeling Language: Superstructure, UML Superstructure Specification v2.0, formal/05-07-04, Object Management Group, 2005
3. F.S. Hsieh and J. BonLin, Development of context-aware workflow systems based on Petri Net Markup Language, *Computer Standards & Interfaces*, vol. 36, no. 8, pp. 672-685, 2014.
4. B. Bousetta, O. Beggar, and T. Gadi, A methodology for CIM modelling and its transformation to PIM, *Journal of Information Engineering and Application*, vol. 3, no. 2, pp. 1-21, 2013.
5. Y. Rhazali, Y. Hadi, and A. Mouloudi, Model Transformation with ATL into MDA from CIM to PIM Structured through MVC, *Procedia Computer Science*, vol. 83, pp. 1096-1101, 2016.
6. V. De Castro, E. Marcos, and J. M. Vara, Applying CIM-to-PIM model transformations for the service-oriented development of information systems, *Information and Software Technology*, vol. 53, no. 1, pp. 87-105, 2011.
7. N. Prat, J. Akoka, and I. Comyn-Wattiau, An MDA approach to knowledge engineering, *Expert Systems with Applications*, vol. 39, no. 9, pp. 10420-10437, 2012.
8. R. Q. Yu, Z. Q. Huang, L. Wang et al., Analyzing BPMN with Extended Object Petri Net, *Journal of software engineering*, vol. 8, no. 2, pp. 58-74, 2014.
9. W. M. P. van der Aalst and A. H. M. ter Hofstede, Verification Of Workflow Task Structures: A Petri-net-baset Approach, *Information Systems*, vol. 25, no. 3, pp. 43-69, 2000.
10. Y. Ye, Z. B. Jiang, X. D. Diao et al., Extended event-condition-action rules and fuzzy Petri nets based exception handling for workflow management, *Expert Systems with Applications*, vol. 38, no. 9, pp. 10847-10861, 2011.
11. J. Boubeta-Puig, G. Díaz, H. Macià et al., MEdit4CEP-CPN: An approach for Complex Event Processing modeling by Prioritized Colored Petri Nets, *Information Systems*, vol. 81, no. 3, pp. 267-289, 2017.
12. C. Dechsupa, W. Vatanawood, and A. Thongtak, Hierarchical Verification for the BPMN Design Model Using State Space Analysis, *IEEE Access*, vol. 7, no. 1, pp. 16795-16815, 2019.
13. C. Dechsupa, W. Vatanawood, and A. Thongtak, Transformation of the BPMN Design Model into a Colored Petri Net Using the Partitioning Approach, *IEEE Access*, vol. 6, no.7, pp. 38421-38436, 2018.
14. R. M. Dijkman, M. Dumas, and C. Ouyang, Semantics and analysis of business process models in BPMN, *Information and Software Technology*, vol. 50, no. 12, pp. 1281-1294, 2008.
15. P. V. Gorp and R. M. Dijkman, A visual token-based formalization of BPMN 2.0 based on in-place transformations, *Information and Software Technology*, vol. 55, no. 2, pp. 365-394, 2013.
16. G. S. Kang, L. Q. Yang, and L. Zhang, Verification of behavioral soundness for artifact-centric business process model with synchronizations, *Future Generation Computer Systems*, vol. 98, no. 2019, pp. 503-511, 2019.
17. P. Felli, M. D. Leoni, and M. Montali, Soundness Verification of Decision-Aware Process Models with Variable-to-Variable Conditions, In *Proceedings of the 19th International Conference on Application of Concurrency to System Design (ACSD)*, pp. 82-91, 2019, DOI: 10.1109/ACSD.2019.00013.
18. M. Foughali and P. E. Hladik, Bridging the gap between formal verification and schedulability analysis: The case of robotics, *Journal of Systems Architecture*, vol. 111, no. 12, pp. 101817, 2020.
19. B. Aristyo, K. Pradityo, T. A. Tamba et al., Model Checking-based Safety Verification of a Petri Net Representation of Train Interlocking Systems, In *Proceedings of the 57th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pp. 392-397, Nara, Japan, 2018.
20. R. Mrasek, J. Mülle, and K. Böhm, A new verification technique for large processes based on identification of relevant tasks, *Information Systems*, vol. 47, no. 1, pp. 82-97, 2015.
21. I. Trickvoic, Formalizing activity diagram of UML by Petri nets, *Journal of Mathematics*, vol. 30, pp. 161-171, 2000.
22. R. Eshuis and R. Wieringa. Comparing Petri Net and Activity Diagram Variants for Workflow Modelling-A Quest for Reactive Petri Nets, *Petri Net Technology for Communication-Based Systems*, vol. 2472, no. 4, pp. 321-351, 2003.
23. T. Bouabana-Tebibel and M. Belmesk, An object-oriented approach to formally analyze the UML 2.0 activity partitions, *Information and Software Technology*, vol. 49, no. 9, pp. 999-1016, 2007.
24. T. S. Staines, Intuitive Mapping of UML 2 Activity Diagrams into Fundamental Modeling Concept Petri Net

- Diagrams and Colored Petri Nets, In Proceeding of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, pp.191-200, 2008.
25. M. A. Ameen and B. Bordbar, A Model Driven Approach to Represent Sequence Diagrams as Free Choice Petri Nets, In Proceeding of the 12th International IEEE Enterprise Distributed Object Computing Conference, pp. 213-221, 2008.
 26. N. H. Yang, H. Q. Yu, H. Sun et al., Modeling UML sequence diagrams using extended Petri nets, *Telecommunication Systems*, vol. 51, no. 3, pp. 147-158, 2012.
 27. J. P. Faria and A. C. R. Paiva, A toolset for conformance testing against UML sequence diagrams based on event-driven colored Petri nets, *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 12, pp. 1-20, 2014.
 28. Z. H. Li, X. F. Zhou, and Z. W. Ye, A Formalization Model Transformation Approach on Workflow Automatic Execution from CIM Level to PIM Level, *International Journal of Software Engineering and Knowledge Engineering*, vol. 29, no. 9, pp. 1179-1217, 2019.
 29. Z. H. Li, X. F. Zhou, A. H. Gu et al., A complete approach for CIM modelling and model formalizing, *Information and Software Technology*, vol. 65, no. 9, pp. 39-55, 2015.
 30. D. Steinberg, F. Budinsky, M. Paternostro et al., *EMF: Eclipse Modeling Framework (2nd Edition)*. Addison-Wesley Professional, pp. 104-124, 2008.
 31. PNML Tools, <http://www.pnml.org/tools.php>.
 32. W3C, "Web Service Choreography Interface (WSCI)," Version 1.0 , 2002, www.w3.org/TR/wsci.