International Journal of Computing Technology and Information Security
Vol.1, No.1, pp.30-39, March, 2011

Analyzing and Verifying SOFL Workflows for Web Service Composition

Shao-chong Li, Haopeng Chen and Xi Chen

School of Software, Shanghai Jiao Tong University
Shanghai, 200240, China
lee.shaochong@gmail.com, chen-hp@sjtu.edu.in, april.chenxi@gmail.com

Abstract: *While the different aspects of flexible workflow in the management are still under discussion, number of domain-specific languages for workflow description have been proposed with consensus being formed around a process-oriented language known as WS-BPEL. However, detecting logic errors and conducting the validation of the business process is quite hard due to BPEL does not have formal semantics. This paper aims at validating the feasibility of using SOFL for static analysis of business process. SOFL can be used as a design language for the specification of complex workflows, and SOFL theory provides for powerful analysis techniques which can be used to verify the correctness of workflow procedures. In particular, we focus on how to apply SOFL modeling approach to validate and analyze workflow which is an extension of WfMC's process definition meta-model.*

Keywords**:** W*orkflow, WS-BPEL, SOFL, Static Analysis.*

1. Introduction

Workflow support collaboration and efficient execution of business process, and have received much attention both from industry and research, which mainly focus on automated coordination of business process to reduce cost and flow times [1,2,3] A workflow based on service calls for effective response to changes in business process, and these are widely invoked in the order given in the workflow specification due to the strong focus on description and validation of the consistency between the specification and implementation. Business Process Execution Language for Web Services (BPEL or BPEL4WS) is a language used for the definition and execution of business processes with Web services.

In BPEL, logic of interactions between given services and corresponding environment are described as a composition of communication actions. These communication actions are interrelated by control-flow dependencies expressed through constructs corresponding parallel, sequential, and conditional execution, event and exception handling, and compensation. However, due to the fact that BPEL has no semantic information, business process described by BPEL is not suitable to validate whether the specification satisfies the user's requirements, even some simple logic errors such as dead lock cannot be detected. Besides, a specification described by BPEL contains a fair number of acknowledged ambiguous descriptions that may lead to different interpretations.

As BEPL comes without formal features and formal method has advantages of precise description and verification, it would thus benefit from the use of formal methods as this could provide a workflow to be precisely described, while many researchers focus on integrating formal descriptions into business workflow. However, there are some researches which mainly focused on transforming BPEL into a formal language, and then reviewing and validating the formal language to achieve precisely validation. They ignore that BPEL contains some defects that cannot describe the workflow integrated as a whole. A specification described by BPEL transformed to other formal language may contain the BPEL's flaw. Therefore, in this paper, we propose a new method to more precisely describe the workflow. At first, according the previous research work, we summarize an extensible workflow meta-model which contain the fundamental features of workflow. And then we utilize a formal language, SOFL, which is chosen as transforming targets because it's

structured, object-oriented and formal, to describe the workflow meta-model of workflow as a workflow language. SOFL integrates data flow diagrams, Petri nets, VDM (Vienna Development Method), and the object-oriented approach in a coherent manner for specifications construction [4]. The SOFL theory also provides rigorous review and verification method. In paper [5][6][7], the authors represent the method of formal verification with fault tree analysis[8].

For the workflow is described in SOFL, it is convenient to check consistency and validate workflow using SOFL theory. The formalized diagram, called condition data flow diagram (CDFD) is suitable for describing the architecture of the system, while the formal textual notation is effective in defining the meaning of its components, including processes, data flows, and data stores. In this paper, we purpose a review and verify framework based on existing review and verify SOFL theory. The detail will be discussed in the following sections.

The rest of this paper is organized as follows. The next section discusses related work in this area. Section 3 provides a summary of extensible workflow meta-model, and describes the basic rule of transforming relate workflow activities into SOFL. Section 4 describes how to conduct the formal review and validation of the SOFL module. In the final section, we conclude this paper and discuss some future work.

2. Relate Work

In our previous work [9], we have proposed web service selection and recommendation mechanism based on SOA. We demonstrated a feasible mechanism for multi-QoS constraints and prediction of user's preference in order to acquire better recommendation. However, we mainly consider the non-functional attributes of web service. More recently, we have extended our work and considered the problem of service selection based on functional attributes. In order to achieve this function, the service and service combination must be described as formal style.

With the rapid development of workflow technology, it is very important for workflow developer to describes the workflow specification precisely. Aiming at BPEL which is lack of a formal semantics and contains ambiguities, many researchers have been accomplished to formalize BPEL [10], using automata, process algebra, and Petri nets and so on.

Automata is a public and base model of formal specification for modeling systems[11], which contains a set of states, actions, transitions between states, and an initial state, so it is convenient to describe the workflow. Díaz[12] shows a case study on converting business processes written in BPEL-WSCDL to timed automata. In paper [13], For develops a tool to translate the BPEL specifications to guarded automata. Although automata can well describe the BPEL, in terms of large scale system and its limitation of describing complicated functions, automata's accuracy cannot be guaranteed in this situation.

Process algebras can be also used in describing the workflow. It can be divided into many forms, such as ACP (Algebra of Communicating Processes), CCS (Calculus of Communicating Systems), CSP (Communication Sequential Process) and so on. Wong[14] discusses the workflow model described by AGP. In terms of formal verification technology, Salaün[15] presents a method of verifying business processes based on processing algebras with particularly focusing on their interactions. In paper [16], Salimifard presents the translation rule between BPEL and process algebras. However, process algebras cannot support dynamic process instantiation and correlation set. It also cannot detect the dynamic structure alteration thoroughly.

As Petri nets have rigorous and profound math fundamental, it can be used to analyze and verify workflow precisely. There are many researches on building workflow model based on Petri nets [16]. It is a prevalent method on describing business process using the theory of Petri nets. Paper [17][18][19] describe the translation rule from BPEL to Petri nets. In paper [20], the authors can translate

composition specified in BPEL into CP-nets, which can be analyzed and verified by many developed tools. However, Petri net is still based on graphical notation and its expressiveness is limited for large scale systems. Besides, for those system involving rich data types and high logical complexity specification, Petri nets can't describe precisely for more specific usages

3. Formal Description Of Workflow

The workflow concentrates on how to execute task in a specific order. The Workflow Management Coalition (WfMC) [21] defines a workflow as follows: workflow is concerned with the automation of procedures where documents, information or tasks are passed between participants according to a defined set of rules to achieve or contribute to an overall business goal. As is described in [22]: the workflow process definition specifies which tasks need to be executed and in what order. Multiple workflow processes' behavior can be integrated as a whole workflow.

Modeling workflow is a complex process. The industrial model contains many aspects. We could not build a universal model because different industrial models may have different angles of concern. Therefore, what we need to focus is that the model of workflow need to contain the kernel meta-model of workflow and the other part could be established by special industrial model. The meta-model and industrial model can be associated to work together. So it is crucial to build an extensible meta-model of workflow. We will describe an extensible meta-model of workflow in details.

Workflow meta-model

In order to define the process of workflow, the WfMC has developed a meta-model of workflow which is depicted in Figure 1. The meta-model is the basic element of workflow objects which implement interoperating function between workflow management systems. We proposed to apply for a new workflow meta-model based on the WfMC.

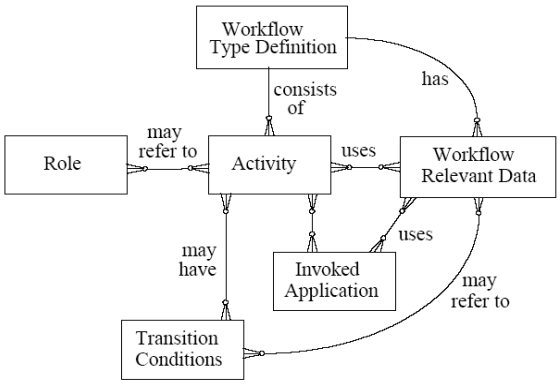

Figure 1. Meta-Model

In terms of workflow's function, workflow can be recognized as the abstract of working process. Thereby, workflow need to contain business process definition, process executed order, pre-and post-conditions, resource and constraint. A meta-model of workflow may contain the kernel elements and extensible mechanism. First, we discuss the kernel elements of meta-model.

Kernel elements: Based on the definition of WfMC, our workflow model contains five kernelparts: task part, resource part, event part, role part and routing constructs part.

Task part: A work task is defined as concrete work item and the basic execution unit of business process which needs to be completed in order to achieve business goals. Each task has pre- and post-conditions: the preconditions should hold before the task is executed, and the post-conditions should hold after execution of the task. A work task with accessing some resource can be completed or executed by users. Each task is executed in a specific routing construction which will be discussed in following. The routing is to identify conditions which correspond to different causal dependencies between tasks.

One task is defined as a work item. It contains the statue, name, input, output, pre- and post-conditions. Typically, a process often is composed of these parts: name, input port, output port, precondition, and post-condition. Here we define a task as the flowing:

Definition1: The elements (Precondition, Post-condition, Role) in task can be defined as a tuple (id, formula, refer), where, *id* the identifier

of element, *formula* is the condition of element's description, *refer* is the id of the element reference Task.

Definition 2**:**Task element can be defined as *TASK***=***(id, in, out, pre, post, role),* where, *id* is the identifier of one task; *in* is the set of input parameters; *out* is the set of output parameters; *pre*, *post* and *role* is a tuple which has been defined as definition 1.

According to the definition 2, one task can be transformed into a process. Figure 2 shows a task described by SOFL.

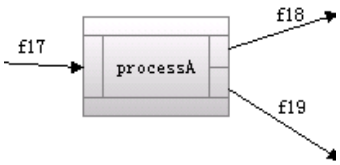

Figure 2. Task

Resource part: A resource contains physical entity which may be accessed by tasks, such as document, data, equipment, etc. In some cases, resource may include person (participant, worker, employee). For example, in most offices, the resources are often defined as people. Different resources are divided into different tasks which is defined by task's attributes. The resource alsocontain its constraints and states. The constraint describes the scope of utilizing resource. State represents whether the resource can be usedor not, therefore, the resource part can be defined as following:

Definition 3: The resource can be defined as *Resource = (id, name, input, output, state),* where, *id* he identifier of resource; *name* is the name of resource; *input* delegates the input resource of data store; *output* represents the output resource of data store; *state* is the conditions of resource which contain write, read conditions and so on. If the resource is people, organization or machine, it will be transformed into an external process as shown in Figure 2, otherwise it will be transformed into data store just like Figure 3.

Figure 3. Resource

Event part: Besides task and resource parts, event part should also be included by the workflow meta-model, because event part describes the contact information between tasks. One event can be defined as the flowing:

Definition 4: A event can be defined as *Event = (id, in, out, from, to)*,where , *id* is the identifier of one event; *in* is the input message of one event;*out* is the output message of one event; *from* is the task which send the message; *to* is the task wihich revive the message.

Event part can be transformed into input and output of one process in SOFL. Besides, data store can also be used to data from message. For example, if one process access to a data store, the data store can be recognized input or output message. Just as Figure 4, the process "Get_Account" has two input message: "user_id" and "password". The data "result" is output message.

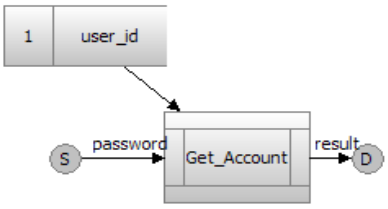

Figure 4. Message

Role part: The role part contains a mechanism which associates the participant with a series of tasks. The important properties contain name, organization entities etc. In SOFL, the role part can be described by a process's property. The definition of Role has been described in the *definition 1*.

Routing construct part: A workflow process definition specifies how the cases are routed along the tasks that need to be executed immediately. According to the routing constructs identification of Workflow Management Coalition (WfMC), there exist four base types of routing: sequential, parallel, conditional, iteration.

Sequential

If the execution of one task is followed by the next task, the tasks are said to be executed sequentially. The sequential activity is used to define a collection of activities to be performed sequentially in lexical order. The executing order in SOFL is determined by the input and output. The Sequential routing is described in SOFL as Figure 5:

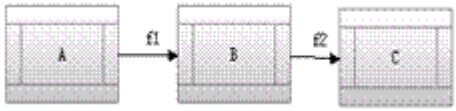

Figure 5. Sequential

Parallel

The parallel routing represent the tasks are executed at the same time or in any order. At the parallel scene, as is depicted in Figure 6, task B and C are executed in parallel style.

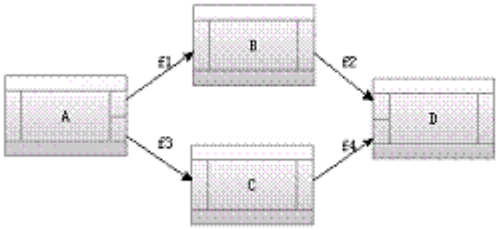

Figure 6. Parallel

Condition

In order to model a choice between two or more alternatives, we use the conditional routing to solve this problem. The condition routing includes if, if-then-, if-then-else structures. Figure 7 shows the branch condition in SOFL:

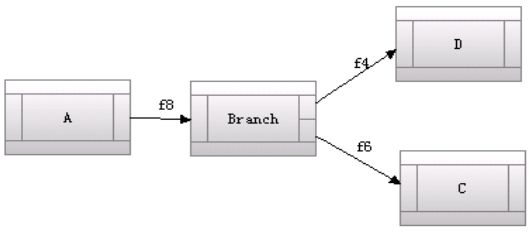

Figure 7. Condition

Iteration

If a task need execute multiple times, it is called iteration routing. Figure 8 depicts this scene.

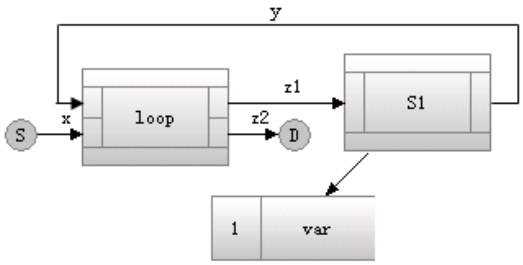

Figure 8. Iteration

Extendible mechanism: In this paper, we consider the process model is the kernel model of workflow. A process model may contain the kernel elements which have described above. Therefore, we can get the kernel meta-model of workflow as shown in Figure 9. The kernel meta-model of workflow is the minimal cut set of workflow. In order to satisfy different industry models, we provide an extendible mechanism to expand the meta-model.

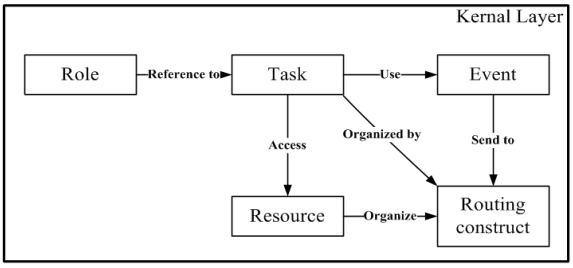

Figure 9. Kernel workflow meta-model

The kernel meta-model defines the objects and theirrelations used in process definition period, however, it is lackof detailed description about corresponding organization andignores the description ofconceptions used in running period.Therefore, besides the kernel layer, we should build a expanding layer to broaden the meta-model of workflow. The modified workflow meta-model is shown in show in Figure 10. According to the Figure 10, for each kernel part, we expand them to satisfy different industry model. Here is detail information about them:

Organization structure and User: The role part can be expanded as organization structure part and user part. In terms of extensible information of workflow, organization structure can provide industry organization information which acquire from industry model. One

organization structure may consist of many roles. User information can be described in user part. One user can be assigned to several roles, and several users can act as one role.

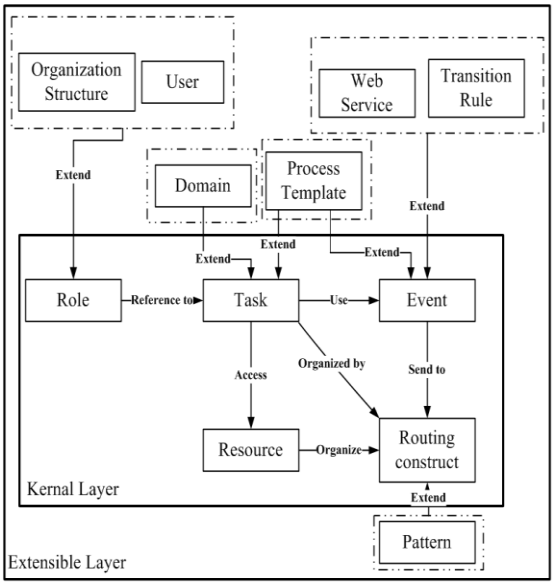

Figure 10. Extensible workflow meta-model

Domain and Process Template: Task part is the base element of workflow. In kernel layer, the task part has no domain information. Therefore, we add the domain part to the task part of workflow meta-model. The domain part format the task to accommodate different industry model. Process template derives from task part and event part. A process template consists of a series of task and a set of transition rules which determines the moving from one task to the next task.

Transition rule and Web service: Event part expands to part: transition rule and web service. Transition rule describes the condition of one task transform to another. Web service represents the new pattern of software architecture. It is often used in workflow to transfer message. This paper expands web service in a new style.

Pattern: In the kernel layer, the routing construct part only represents the base elements of routing. However, in industry model, the routing construct is always complicated, because it composes the basic elements of routing construction. So we provide a pattern part to describe the composed routing. For example, we can extend the "while" pattern based on the basic routing structure.

Therefore, we expand the meta-model into two layers: kernel layer and extensible layer. Kernel layer concentrates on base processes interaction without industry model's information. For the different industry models, it could be used in the extensible layer to expand workflow model.

5. Formal Analysis of Workflow

Web services provide a novel implementation way of loosely coupled Service-Oriented Architecture (SOA) which is widely accepted by the information industry as the future internet application architecture and ease the process of application integrations. In general, SOA and web services are used for creating new services by composing existing services together. The need for inter service compatibility analysis and indirect composition has gone beyond what the existing service composition and verification technologies can handle. For example, a BPEL file may contains the <invoke> tag which relate to a invoke service. However, from the BPEL file, we could not obtain that whether the invoke service can satisfy the author's request. That's because the workflow described by BPEL only depicted the structure of workflow but have no semantic information. Therefore, the logic error of service composition such as dead lock, definition incomplete can't be detected. Besides this, whether the service composition could satisfy the user's requirement cannot be validate.

But for the workflow described by the SOFL, it is convenient to review and validate results of the service composition. The workflow consists of several different services, which need to be reviewed and validated to detect whether satisfy the user's requirement. As we have described above, one process can be decomposed by many CDFDs and their associated modules, checking the service composition is transformed into checking whether the decomposition satisfy the process. We use the relate SOFL theory to validate the following determinant factors of service composition:

- Internal consistency
- Satisfiability
- Deadlock
- Verifiability

Formal Review of workflow

In this section, we use the methods in paper [7] for reference, and propose how to review the workflow described by SOFL.

Internal consistency

Refer paper [7], Process internally consistent:

- Process internally consistent
- Process and invariant consistent
- The adjacent processes are consistent

At first, we define some notation:

- Input(P): the set of all input data flow variables of process P
- Output(P):the set of all output data flow variables of process P
- WR(P): the set of all writable(wr) external variables(including both decorated and undecorated variables) of process P
- RD(P): the set of all readable (rd)external variables of process P.

According to the paper [7] rule, if p satisfies the following rules, we can determine process p is internally consistent:

a) $\boldsymbol{forall}[v: \boldsymbol{Output}(P)] \mid v \ \boldsymbol{notin}\ \boldsymbol{Variables}(pre\_P)$

b) $V\ \boldsymbol{inset\ union}\left(\boldsymbol{Variables}\left(pre_p\right), \boldsymbol{Variables}\left(post_p\right)\right) \Rightarrow V\ \boldsymbol{inset\ union}\ (\boldsymbol{Input}(p), \boldsymbol{Output}(p), \boldsymbol{WR}(p), \boldsymbol{RD}(p))$

c) $\boldsymbol{forall}[x : \boldsymbol{int}] \mid pre\_P\ (x) => \boldsymbol{exists}[y : int\ ]\ \mid \boldsymbol{post\_P}(x, y)$

Using this rule for the workflow described in SOFL, for the consistency of business process, we should keep each variables generated consistently. For example, suppose there exist two processes A and B, and a, b is input and output variable of process A respectively. Process B is ahead of process A. Variable v is in the union set of pre-condition variables and post-condition variables. In order to ensure the output variables generated consistently, we should review:

a) Output variable y is only made variable as the result of executing the process A. It must in the post-condition of process A but not in the pre-condition.

b) If variable v used in the pre- and post-conditions must be one of the input, output, and external variables of the process.

c) If process A and its ahead process B satisfy ***pre_B***and ***post_B***(x) =>***pre_A***, we say that process A is consistent with the process B.

The algorithm of reviewing internal consistence is described in Figure 11.

```
checkConsistency( Process A )
1.  Set_out ← Output(A);
2.  Set_in  ← Input(A) ;
3.  Set_pre ← Variables(Pre(A));
4.  Set_post ← Variables(Post(A));
5.  Set_data ← WR(p) join RD(A);
6.  for each variable in Set_out
7.      if variable in Set_pre   return false;
8.  for each element in join(Set_pre ,Set_post )
9.    if element not in join(Set_in ,Set_out ,Set_data ,)
    then return false;
10.  return true;
```

Figure 11. Reviewing Internal Consistence

Satisfiability

The satisfiability ensures that for any input satisfying the pre-condition, there exists an output meeting the post-condition, based on the input. Refer paper [26], the satisfiability of process P is defined as:

$\forall \boldsymbol{a, y}, \tilde{}\boldsymbol{z}: \boldsymbol{UsableInt}\ \boldsymbol{pre\_P}(\boldsymbol{a, y, z})[\tilde{}\boldsymbol{z}/\boldsymbol{z}] \Rightarrow \exists\ b, x,$
$\exists\ b, x, z: \boldsymbol{UsableInt}\ \boldsymbol{post\_P}(a, b, x, y, \tilde{}z, z)$

For reviewing the satisfiability of workflow described in SOFL, it should review:

a) For any variable a in the pre_P, there must exist a correlate variable b in the post_P

b) For any input, if the pre-condition evaluates true, there must exist an output based on which the post-condition evaluates true.

The algorithm of reviewing satisfiability is described in Figure 12.

```
checkSatisfiability(Process A)
1.  Set_pre ← Variables(Pre(A));
2.  Set_post ← Variables(Post(A));
3.  for each element in Set_pre;
4.      if A.relatefunction(element) in Set_post;
            then  return true;
5.    else return false;
```

Figure 12. Reviewing Satisfiability

Deadlock

Reviewing the workflow described by no-formal language is a difficult even impossible task. But for the workflow described by SOFL, this task becomes conveniently. The workflow's deadlock is described as this: "In the business process, if process A waits for the result of process B, and process B also waits for the result of process A at the same time." In order to review deadlock of workflow, we need to review whether the adjacent processes of a module exist deadlock. Here we purpose the following steps to review deadlock of workflow:

a) For the adjacent processes A and B, if the input of process A is B and the output of the process A is input of process A, there will exist a deadlock between A and B.
b) If the adjacent processes A and B are not existing deadlock, we need review whether B and B's adjacent process existing deadlock. At the same time, we also review whether A and B's adjacent process exist deadlock. According to such rule, for each process of a module, we need to review whether it and its former process include deadlock.
c) After step *a)* and *b)*, if it can't review deadlock, we may consider that a business process doesn't exist deadlock.

The algorithm of reviewing deadlock is described in Figure 13.

```
checkDeadLock(Process A, Process B)
1.  if(A.getPre() == B.getPost())
        then return true;
2.  else return checkDeadLock(A,B.getNextProcess())
```

Figure 13. Reviewing DeadLock

Formal Validate of workflow

Although the above review methods can detect logic error, they could not detect the semantic errors. It is important to validate whether the business process satisfies the requirements of the users. As the process in SOFL is described with precondition and post-condition, we can compare the precondition and post-condition to verify the consistence between business process and requirements of the users. If a process has decomposition, we also need to validate the consistence between process and CDFDs from decomposition. Paper [6] has brought forward the method of reviewing validity of SOFL. For the workflow described by SOFL, a complex workflow is composed by many processes, data flows, and data stores as well as other structures. As a process can be decomposed many CDFDs and their associated modules, we need to validate whether the decomposition satisfy the process.

The detailed arithmetic is described as following:

At first, the whole SOFL file will be compiled as a semantic tree. For example, suppose we have the SOFL specification as following:

```
1.  module SYSTEM_simple;
2.  behav CDFD_simple;
3.  end_behav;
4.  process Init()
5.  end_process;
6.  end_module;
```

Figure 14. Module

We built a semantic tree as Figure 15.

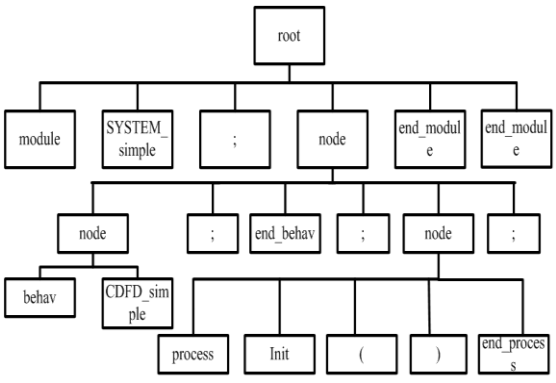

Figure 15. Semantic Tree

And then, from the top module of the semantic tree to the bottom, we need validate each process with decomposition. And then, we will verify the process according to the following rules:

a) If each process in the module has no decomposition, we will consider this module need n't to be verified.
b) For each process in the module, if it has decomposition, we need verify whether the process satisfies its decomposition.

c) According to the rules of paper [12], one process could decompose into different CDFDs. Such as branch, loop, parallel and so on. Therefore, each CDFD could be described by a predicate expression which contain pre and post condition. And then, we could validate process using following rules:

$$\boldsymbol{pre}(S) \Rightarrow \boldsymbol{pre}(P) \quad (1)$$

$$\boldsymbol{pre}(S) \text{ and } \boldsymbol{post}(P) \Rightarrow \boldsymbol{post}(S) \quad (2)$$

S indicates a process, P indicates S' decomposition. We use the above rule to verity process satisfies its decomposed CDFDs.

d) The whole validating procedure is a recurrence pattern. After finding a first requisite validate process from top to bottom, we need verify the process from bottom to top.

In the end, if all the processes are satisfied of their decompositions, we will consider the workflow described by SOFL has satisfied user's requirement. Otherwise, it is not satisfied.

Step (c) is a complex approach which needs a formal method to prove the consistence between process and its decomposition. In order to validate process using step c) rules, we need to transform the predicate expression of precondition and post-condition to disjunctive normal form. For example, P => Q will be transformed into !P or Q. Refer paper [8], they use RTT and minimal cut sets to transform the predicate expression. The detailed steps are described as following:

a) After the semantic tree has been established, we built a RTT to form the expression which only contain "and","!" and" or" operators.
b) Once RTT has been built, we can compute the minimal cut sets which are a set of atomic operations to simplify the disjunctive normal form.

The next example will show the process of transformation. Suppose we get the predicate expression from semantic tree: !(x > 0 and x < ~w) or (y = x + ~w or z = x - ~w), and we can build RTT as Figure 16.

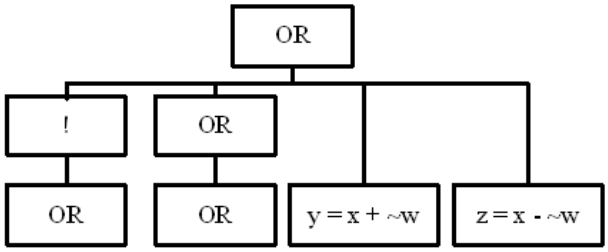

Figure 16. RTT

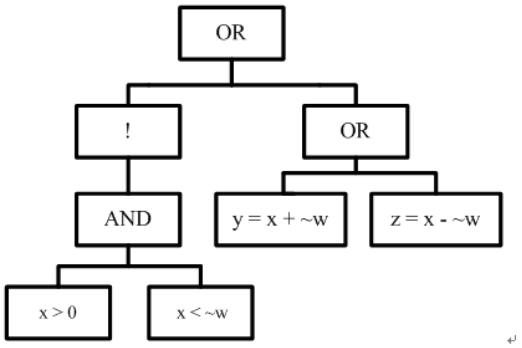

Figure 17. Example

And then we can get the minimal cut sets from RTT. There are many researches about arithmetic of minimal cut sets. Here we purpose an example as Figure17. From the Figure 17, we can get the minimal cut sets: {!(x>0)}, {!(x<~w)}, {y=x+~w}and{z=x-~w}. At the end, we get the disjunctive normal form !(x > 0) or !(x < ~w) or (y = x + ~w) or (z = x - ~w).

Once we get the normal form, we can prove whether the decomposed CDFD normal form satisfies the module. The validate method uses formula (1) and (2). Truth table or other mathematical methods can be used to proof equivalent between two disjunctive normal forms.

5. Conclusions and Future Work

In this paper, the workflow meta-model is a modification and extension of WfMC's process definition meta-model. Therefore, the model becomes active, adaptive, and customizable. Based on SOFL to describe the workflow meta-model, we can thoroughly resolve the problem of service composition. In this paper, we utilize SOFL related review and validate

theory to verify the adaptability of service composition.

However, in the validate step, we have to prove whether the disjunctive normal forms are equivalent in the following research work. This may associate the theory of discrete mathematics. Our future work will concentrate on how to prove the equality between two disjunctive normal forms.

References

[1] K.Hayes and K. Lavery, "Workflow Management Software:The Business Opportunity", Technical report,Ovum Ltd, London, UK, 1991.

[2] S.Jablonski and C. Bussler, "Workflow Management: Modeling Concepts, Architecture, and Implementation", International Thomson Computer Press, London, UK, 1996

[3] T.M. Koulopoulos, "The Workflow Imperative", Van NostrandReinhold, New York, USA, 1995.

[4] Shaoying Liu, "Formal Engineering Methods for Information Systems Development", in Proc. of Second International Conference on INFORMATION, Beijing, pp. 148-154, 2002.

[5] Shaoying Liu, "A Property-Based Approach to Reviewing Formal Specifications for Consistency", in Proc. of 16th International Conference on Software & Systems Engineering and Their Applications, Paris, France, Vol. 4, pp.1-6, 2003.

[6] Shaoying Liu, "An Automated Rigorous Review Method for Verifying and Validating Formal Specifications", in Proc. of Second International Symposium on Automated Technology for Verification and Analysis, LNCS 3299, Springer-Verlag, Taipei, Taiwan, pp.15-19, 2004.

[7] Shaoying Liu, "A Rigorous Approach to Reviewing Formal Specifications", in Proc. of 27th Annual IEEE/NASA International Software Engineering Workshop, IEEE Computer Society Press, pp. 75-81, 2002.

[8] Shaoying Liu, "Utilizing Specification Testing in Review Task Trees for Rigorous Review of Formal Specifications", in Proc. of Asia-Pacific Software Engineering Conference (APSEC03), pp.510-519, 2003.

[9] Shaochong Li and Hao-peng chen, "A Mechanism for Web Service Selection and Recommendation Based on Multi-QoS Constraints", in Proc. of WS-CS-Testing, pp.205-211, 2010.

[10] Shoichi Morimoto, "A Survey of Formal Verification for Business Process Modeling", Lecture Notes in Computer Science, Vol.5102, pp. 514-522, 2008.

[11] Hopcroft JE, Motwani R, and Ullman JD, "Introduction to Automata Theory, Languages, and Computation", 3rd Ed., Addison-Wesley: Reading, 2006.

[12] Díaz G, Pardo JJ, Cambronero ME, Valero V, and Cuartero F, "AutomaticTranslation of WS-CDL Choreographies to Timed Automata", Lecture Notes in Computer Science, Vol.3670, pp.230-242, 2005.

[13] Fu X, Bultan T, and Su J (2004), "Analysis of Interacting BPEL Web Services". In Proc. of the 13th International Conference on the World Wide Web (WWW 2004) pp. 621-630, 2004.

[14] Wong K F, Low B T and Yongjie R, "A workflow model for Chinese business processes", International Journal of Computer Processing of Oriental Languages, Vol.14, No.3, pp.233-258, 2001.

[15] Salaün G, Bordeaux L, and Schaerf M, "Describing and Reasoning on Web Services using Process Algebra", in Proc. of the InternationalConference on Web Services (ICWS 2004), pp.43-50, 2004.

[16] Salimifard K and Wright M., "Petri net-based modelling of workflow systems: an overview", European Journal of Operational Research, Vol.134, No.6, pp. 64-676, 2001.

[17] H.M.W. Verbeek, and W.M.P. van der Aalst, "Analyzing BPEL processes using Petri nets", in Proc. of 2nd International Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management, pp.59-78, 2005.

[18] W.M.P. van der Aalst, "Verification of workflow nets", in Proc. of 18th International Conference on Application and Theory of Petri Nets, Lecture Notes in Computer Science, Springer-Verlag, Vol. 1248, pp.407–426, 1997.

[19] W.M.P. van der Aalst, M. Dumas, C. Ouyang, A. Rozinat, and H.M.W. Verbeek, "Choreography conformancechecking: An approach based on BPEL and Petri nets", (extended version), Technical Report BPM-05-25, BPMcenter.org,2005.

[20] Y.Yang, Q.Tan, J.Yu and F.Liu, "Transformation BPEL toCP-Nets for Verifying Web Services Composition", in Proc. on the International Conference on NextGeneration Web Services practices (NWeSP'05), pp.407-415, 2005.

[21] David Hollingsworth, "Workflow Management Coalition: The Workflow Reference Model" (WfMC-TC00-1003 Issue 1.1),1995.

[22] W.M.P.van der Aalst. "The Application of Petri Nets to Workflow Management", The Journal of

Circuits, Systems and Computers, Vol.8, No.1,
pp.21- 66, 1998.