http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 8 (7): 998-1005, 2009
ISSN 1812-5638

A Process Generation Approach of Dynamic Workflows Based Description Logics

[1]FuXin Zhang and [1,2]YuYue Du
[1]College of Information Science and Engineering,
Shandong University of Science and Technology, Qingdao 266510, China
[2]The State Key Laboratory of Computer Science, Institute of Software,
Chinese Academy of Sciences, Beijing 100080, China

Abstract: To make workflow processes more flexible, a dynamic generation approach for workflow processes corresponding to an instance is presented in this study. An activity, a part of a workflow, is defined as an action based on Description Logics (DLs). User preferences are considered, since a final solution should satisfy user preferences as much as possible. Also, a hierarchical workflow ontology model is proposed and a deciding method for basic routing relations in workflows is provided to produce the processes completely. Then for generating the processes, a new planning algorithm of workflow processes, DPWPG: Dynamic Planning for Workflow Process Generation, is presented and used to search matching activities in a workflow ontology model, according to ontological reasoning in semantic activities and users' preferences. Finally, an example is given to test the performance of the planning algorithm.

Key words: Semantic activity, description logics, workflow ontology, AI planning

INTRODUCTION

A workflow management technology aims at the automated support and coordination of business processes to reduce cost and flow time and enhance quality of services and productivity (Van der Aalst and Kees van Hee, 2002). In accordance with the rules defined in advance, a series of business activities will be composed and executed. However, the real run-time processes are often much more variable than the processes specified at design-time (Sadiq *et al*., 2001; Heinl *et al*., 1999). The workflow cannot be presented from complete processes and logical relationships between activities. The specific path of a current instance can be determined only in the run-time of a workflow instance, according to the context. Therefore, the ability of responding effectively to the change is a critical challenge for workflow management systems which do not support dynamic change of business processes.

To deal with the change in workflow systems, the notions of pockets of flexibility and black boxes in workflow specifications are introduced in some papers (Sadiq *et al*., 2001; Heinl *et al*., 1999). A dynamic change approach for workflow management is described by Rinderle *et al*. (2004). This approach is based on a migration rule, which dictates how an instance will be migrated from an old workflow process to a new workflow process based on the change of an execution environment.

Recently, an important method with the ability of intelligible computing and semantic technology is mostly used to improve the capability of workflows. Workflow ontology can describe the relationships between workflow components, such as processes, activities and roles, for automatic reasoning about a dynamic workflow process. In fact, a workflow ontological model can represent knowledge by means of a frame structure. A workflow ontology named DAML-W has been defined based on a DAML Language (Wu *et al*., 2005). To construct a semantic oriented workflow mechanism, furthermore, a methodology is proposed based on Multi-Agent systems. Ontology based workflow knowledge meta-model is presented and supports the composition of business processes (Han *et al*., 2007). A mechanism proposed in allows a workflow execution to proceed in the presence of incomplete information by using workflow ontology (Vieira *et al*., 2004). To implement the web service discovery and composition in run time of a workflow, an agent based workflow ontology model is given and it provides a better flexibility for the reasoning and coordination of workflow processes (Wang *et al*., 2005). Action formalism is introduced for the

Corresponding Author: YuYue Du, College of Information Science and Engineering,
Shandong University of Science and Technology, Qingdao 266510, China

reasoning of an action community based on Description Logics (DLs) (Baader *et al*., 2005). To automatically combine component workflow instances, an ontology-based reasoning method is presented and shows an ontology-driven architecture for a flexible workflow execution (Korhonen *et al*., 2003).

In order to make workflow processes more flexible, in this study, a generation method of the processes in dynamic workflows is investigated. The process corresponding to a current instance can be constructed in the run-time of a workflow, which significantly increases processes adaptability.

BASIC FORMALISM OF DESCRIPTION LOGICS

Description Logics (DLs) are a family of knowledge representation languages used to represent the knowledge of an application domain in a structural and formal well-understood way (Baader *et al*., 2007; Sirin, 2006). A knowledge base comprises two components, TBox and ABox. TBox represents the terminology i.e., a vocabulary of an application domain, while ABox contains the assertions named individuals in terms of the vocabulary. Elementary description consists of the set of atomic concepts N_C and the set of atomic roles N_R. The concepts denote sets of individuals and roles denote a binary relationship between individuals, R (a, b), where R a role name and a, b is the name of two individuals respectively. An ABox assertion is of the form C (a), where C is a concept.

In addition to atomic concepts and roles, DLs can build a complex description of concepts and roles on the base of constructors. Usually description logics contain at least the following logical constructors: conjunction $\cup$, disjunction $\cap$, negation $\neg$, exit quantifier $\exists$, Universal quantifier $\forall$. The DLs allowing for negation, conjunction and value restrictions is called ALC. The syntax and semantic of ALC are shown in Table 1.

The semantics of ALC-concepts and roles is defined in terms of an interpretation $I = (\Delta^I, \bullet^I)$. A domain Δ^I of I is a non-empty set of individuals and an interpretation function $\bullet^I$ maps each concept name A to a subset A^I, each role name R to the subset of $\Delta^I \times \Delta^I$, which is called the interpretation of an atomic concept A and an atomic role R.

The concepts and domain knowledge can be defined based on the set of the constructors in Table 1, starting with a set of concepts names, a set of role names. The additional constructors could be added to ALC, which are an extension of ALC and strengthen the expressive power of DLs (Baader *et al*., 2005; Sirin, 2006). However, in the paper, the proposed reasoning framework of semantic activities is restricted to ALC.

Table 1: Syntax and semantics of ALC

Name	Syntax	Semantics
Concept	A	$A^I \subseteq \Delta^I$
Role	R	$R^I \subseteq \Delta^I \times \Delta^I$
Conjunction	$C \cap D$	$C^I \cap D^I$
Disjunction	$C \cup D$	$C^I \cup D^I$
Exit restriction	$\exists RC$	$\{a \in \Delta^I \mid \exists b\ (a, b) \in R^I \wedge b \in C^I\}$
Value restriction	$\forall RC$	$\{a \in \Delta^I \mid \forall b\ (a, b) \in R^I \wedge b \in C^I\}$
Negation concept	$\neg C$	$\Delta^I \backslash C^I$

SEMANTIC ACTIVITY

In the field of AI, the implementation of actions leads to a change in a world state. The definition of actions is used to describe the dynamic changeable processes of the world. An action can be described by a list of pre-conditions and post-conditions. The pre-conditions have to be fulfilled before an action becomes active and post-conditions represent the expected effects of an agent's action.

In the same way, an execution of an activity in a workflow can also make the world change from a state of the environment to another. For this purpose, activities are described as the actions with pre-conditions and post-conditions. The conditions are expressed based on DLs assertions and a current state of the world is described by a set of ABox assertions. The activity defined in DLs is called a semantic activity. A semantic activity is defined formally as follows:

Definition 1: (Semantic Activity:) Let T be an acyclic TBox. For a semantic atomic activity $a \equiv (P_a, E_a)$ in T consists of: a is the name of semantic activities.

P_a = {Pre, Inp}, the finite set of the preconditions and input parameters of the activity, both of which are in the form of ABox assertions, where:

Pre are the preconditions of the semantic activity. Pre specify things that must be true so that the current activity can be executed by an agent;
Inp are the set of input parameters. Inp represent data flows corresponding to the workflow instance. Each assertions in P_a must hold before the execution of the activity a;

E_a = {Eff, Out}, the set of effects and the output parameters of the activity, both of which are in the form of ABox assertions, where:

Eff are the effects of the semantic activity. Eff characterize the side-effects of the activity after the execution;
Out are the set of output parameters. Out denote the modifications to the knowledge base as a result of the execution. Each assertion in E_a will hold after execution of the activity a;

Definition 2: (Execution of semantic activities): Let T be an acyclic Tbox, a, an activity in T with a = (P_a, E_a). I is an interpretation of T. a is executable in A w.r.t. T with I→a I', iff interpretation I satisfies a set of P_a and I | = P_a iff I satisfies P_a.

Definition 3: Let T be an acyclic TBox, a, an activity in T with a = (P_a, E_a), I→a I' for interpretations I and I' of T. If C is a primitive concept and R is a role name, then I' can be got in the following way:

$$C^{I'}: = (C^I \cup \{ b^I \mid C (b) \in E_a, I \mid = P_a\})$$
$$R^{I'}: = (R^I \cup \{(a, b)^I \in E_a, I \mid = P_a \})$$

Since, the interpretation of the defined concepts is uniquely determined by the interpretation of the primitive concepts and the role names, it follows that there cannot exist more than one I' such that I→a I'.

Definition 4: (Execution of compound semantic activities): Let T be an acyclic TBox, a_1,a_2,…a_n n compound activities in T with a_i = (P_{ai}, E_{ai}), a_1,a_2,… and a_n are executable for interpretations I and I' of T, notation I→a_1,a_2,…,a_n I', iff the following conditions are satisfied.

(1) I |= P_{a1};
(2) For 1 ≤i≤n, ∀I^k: I → a_1, a_2 …, a_k I^k, I^k|= P_{ak+1}.

Definition 5: (Activity network): W = (List, Δ) is an activity network, where:

- List is an activity list (a_1, a_2,…, a_m) and a_i is a semantic activity, 1 ≤i≤n.
- Δ is a set of the activity-ordering constraints of the activities in list.

Activity network is used to express the routing relations among activities. There is a set of routing relations in workflows, such as sequence, or-join, or-split and-split and-join and iteration. Assume that routing relations or-join and or-split do not exist in the method introduced in this paper. Other routing relations in Δ can be expressed as follows:

- A sequential relation is expressed by form (a_i<a_j), where activity a_i must be accomplished before activity a_j, i.e., a_i is an immediate predecessor of a_j and a_j is an immediate successor of a_i.
- An and-split relation is expressed by form ((a_i<a_j), (a_i<a_k)), where a_j and a_k must be executed after a_i has been completed and a_j and a_k are executed in parallel.
- An and-join relation is expressed by form ((a_i<a_k), (a_j<a_k)), where a_k synchronizes two parallel activities a_i and a_j.
- An iteration relation is expressed by form ((a_i<a_j), (a_j<a_k), (a_k<a_i)), where a_i is the first activity in next cycle after a_k is executed.

WORKFLOW ONTOLOGY

Ontology is a set of terms for describing a domain that can be used as a skeletal foundation for a KB. It includes a set of concepts within a domain and the relationships between those concepts. By building up the workflow ontology, the properties and capabilities of the workflow can be encoded in an unambiguous, machine-understandable form. Furthermore, the component of the workflow can be automatically interpreted in run time to generate a workflow process which is also shared and reused by groups of other users. A workflow generally has a hierarchical structure and a three-level workflow ontology is proposed as shown in Fig. 1.

In Fig. 1, a workflow is comprised of the abstract activity and the execution process at the workflow ontology level. The abstract activity and the execution process are decomposed into a set of execution activity at the process ontology level. At the activity ontology level, the execution activity is provided by the atomic activity or compound activity. The reason that the workflow is decomposed into abstract activity and execution process is that in the workflow process, there usually exist one or more change regions in which the sequence of activities is variable for diverse instances (Sadiq *et al*., 2001). The change regions tend to accomplish one or more goals. The activities of which the sequence is nondeterministic in the change regions are viewed as an abstract activity in this study. Meanwhile, a part of or a whole workflow process may follow a fixed order for every instance, as a

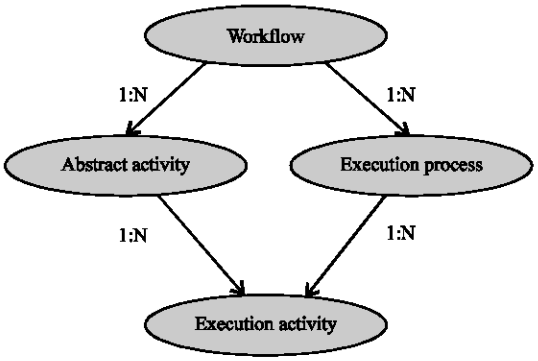

Fig. 1: The construction of workflow ontology

result, the part of or the whole process is described as an execution process. As shown at Fig. 1, our workflow ontology entities are defined as follows:

Abstract activity: An abstract activity is an abstraction of a series of atomic activities accomplishing one or more goals. It is not considered to be directly executable, but provides an abstract view of a workflow process. An abstract activity provides the properties of an activity such that: activity preconditions and effects, activity input and out parameters. Activities are integrated in run time according to the state of a current instance and correspondingly a situation specific, instantiated workflows process will be produced eventually.

Execution process: An execution process is a concrete workflow ontology that describes a detailed workflow process. The sequence of the activities in the execution process is invariable for every instance. An execution process is composed into execution activities without the abstract activity. The sequence of the activities in the execution process has already been determined at build time. Therefore, a execution process can be directly executed in run time without the logic reasoning procedure.

Execution activity: An execution activity inherits the properties of the abstract activity. It is an atomic activity, i.e., the semantic activity defined earlier. An atomic activity with its preconditions satisfied and input parameters bound to particular values is a model of a single step activity that is directly executed to accomplish some goals.

By building up the workflow ontology, a local KB of workflow can be formed. The local KB is populated with the properties and capabilities of semantic activities for knowledge-based indexing and retrieval of activities. In addition to the workflow ontology, a variety of semantic descriptions of user preferences are introduced. Gerevini and Long (2005) describes user preferences in the language PDDL3 and (Son and Pontelli, 2004) presents a language, PP, for the specifications of user preferences. However, in the study, the description of user preferences is based on DLs to customize the user request for the workflow process generation. The DLs have the ability to express the user preferences, which could make user preferences machine-understandable. Figure 2 shows the basic components of workflow system framework. It is composed of workflow ontology, user preferences and workflow engines.

Workflow engine: A workflow engine is responsible for sending requests to workflow ontology for appropriate semantic activities and dispatching activity responses back to the current workflow instance. To achieve the current instance goals or sub-goals, a workflow engine decomposes an abstract activity into execution process by reasoning about the activities in the knowledge base. When a workflow instance is triggered, a workflow engine will interpret the workflow process from high-level ontology. For abstract activity, the engine finds the matched activities for the current instance from the execution activity ontology and coordinates the execution of the workflow process. However, a workflow engine just monitors the consistency among activities for the execution process without runtime reasoning. There may

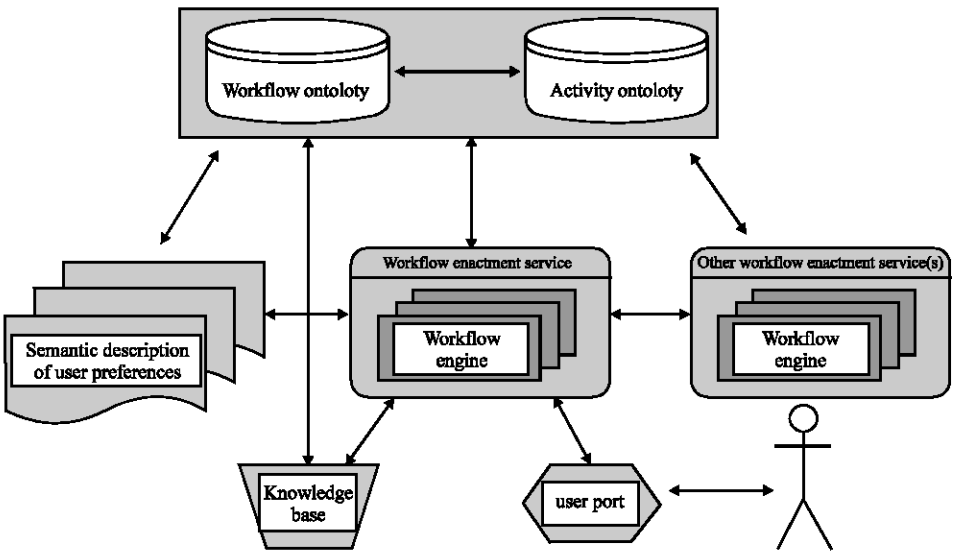

Fig. 2: A framework for a workflow system

be more than one workflow engine in the workflow system. The study is distributed to multiple engines so that the efficiency of the reasoning will be improved.

Semantic description of user preferences: The aims of the dynamic workflow introduced in this study are to generate a solution that not only achieves some absolute goals associated with the instance, but also is desirable with respect to user-provided preferences. Examples of user preferences include contractor 1's ordering plan, which he or she prefers to accept an order from the manufacture 1. If the preference is violated, it does not affect the correctness of the process generation but specifies that certain process is more preferable than others. With description logics, the information necessary for user preferences can be specified as computer-interpretable semantic knowledge and a workflow engine can automatically locate the appropriate activity. A more compact representation of the constraint with description logics is: accepted (contractor 1, manufacture 1), which can use deductive reasoning to infer which activity is the most suitable one.

DECIDING ROUTING RELATIONS AMONG SEMANTIC ACTIVITIES

The workflow management coalition (WFMC) defines four types of routing constructs specifying how instances are routed along the activities that need to be executed: Sequence and-split and-join, or-join, or-split, iteration. With regard to the activity planning method described in this study, it is necessary to indicate the routing relations among activities. The following shows the methodology that takes the description logic-based semantic activities as above and decides the routing relations among them. For routing or-join and or-split, because during planning, the activity, the part of the current instance process, is determined, the routing or-join and or-split do not exist. The deciding of the iteration routing can not be handled currently. The study will be done in the future to address how to decide the iteration routing relations. The deciding methods for other routing relations are shown as follows:

Sequential routing deciding: Let T be an acyclic Tbox, a_i, a_j, two activities in T, for any interpretation I of T, if there exists an interpretation I' such that $I \rightarrow a_i I'$, $I \not\models P_{aj}$ and $I'_i \models P_{aj}$, the execution of a_j must be followed by a_i. Similarly, for any interpretation I of T, if there exists an interpretation I'_i of T, such that $I \rightarrow a_j I'_i$, $I \not\models P_{ai}$ and $I'_i \models P_{ai}$, the execution of a_i must be followed by a_j.

And_split routing deciding: Let T be an acyclic Tbox, a_0, a_i, a_j, three activities in T, for any interpretation I_0 of T, if there exists an interpretation, I'_0 of T, such that $I_0 \rightarrow a_0 I'_0$, $I'_0 \models P_{ai}$ and $I'_0 \not\models P_{aj}$, there also exists an interpretation I'' such that $I'_0 \rightarrow a_i, a_j I''$ and $I'_0 \rightarrow a_j, a_i I''$, activities a_i and a_j are executed in parallel, besides, a_i, a_j must be executed after a_0 has been completed.

And_join routing deciding: Let T be an acyclic Tbox, a_0, a_i, a_j three activities in T, for any interpretations I_i, I_j, if there exists interpretations I'_i, I'_j of T, such that $I_i \rightarrow a_i I'_i$, $I_j \rightarrow a_j I'_j$, $I'_i \not\models P_{a0}$, $I'_j \not\models P_{a0}$, but $I'_j \cup I'_j \models P_{a0}$, a_0 synchronizes the two parallel activities a_i, a_j.

The above methods are shown to decide the execution sequence among activities. Only the sequence of activities is determined, the planning algorithms can generate a solution which meets the needs of the currently running instance.

DPWPG: DYNAMIC PLANNING FOR WORKFLOW PROCESS GENERATION

A planning algorithm, DPWPG: Dynamic planning for workflow process generation is described here, in order to generate dynamically a workflow process satisfying the current instance preferences. The following shows the description of a DPWPG planning algorithm. The DPWPG algorithm is based on backward chain, which can eliminate redundant activities that are not relevant to the instance during the planning process. The core idea of the algorithm is described as follows:

The effects and out parameters of the abstract activity can be got at first, which are regarded as the ultimate goal of the planning. During the planning process, the ultimate goal is decomposed into several sub-goals recursively. Then, the description logic-based reasoning can be used to get the activities realizing the sub-goals. According to the definition of the semantic activity, P_a will turn into the goals of the next planning. Note that the activities at the same layer should be executed in what order to achieve the sub-goals is very important in the planning process. By the methods proposed earlier, the sequence can be determined. After the planning for sub-goals at the same layer is completed, the planning for following sub-goals will be in process.

The description of the planning algorithm is shown below. The workflow process must be deductively planned in the context of the domain knowledgebase, which includes activity ontology and the properties of the user. The algorithm takes four inputs I_0, A_{abs}, K_{ont}, P_{mark} and W. I_0 is the initial state, i.e., the initial interpretation, A_{abs}

is an abstract activity, K_{ont} is the domain knowledge including the activity ontology and process ontology, P_{mark} is semantic description of user preferences, W is an activity network.

The algorithm first gets ultimate goals from an abstract activity. The extract subroutine is responsible for the operation.

The matching procedure of activities should take into account the users' preferences, using deductive machinery, to customize a business process. In the algorithm, the Matching subroutine searches for the activity which can achieve goal in the K_{ont}. At the same time, the matching checks the preferences in the P_{mark} and verifies if the currently searched activity entails users' constraints. If the activity satisfies users' preferences, the algorithm puts the activity into activity list. Otherwise, the matching searches for an activity in the K_{ont} whose effect satisfies user preferences as much as possible. Once a workflow process planning is finished, the process can be achieved in a sharable workflow process so that multiple users with the same constraints can access it directly.

```
Available_goal: current goal set
Available_result: matching results set
DPWL (I0, Aabs, Kont, Pmark, W)
{
Available_goal ← extract (Aabs)
Repeat
{
For each goal in the available_goal
{
Select the goal from Available_goal and remove it
List ← matching (goal, Kont, Pmark))
}
If the final activity a1…, an can accomplish the current
goals in the set Available_goal then
{
For (i = 1 to n)
If (∃ai, I0|≠ai)
Then
{
Put Pai into the set available_goal and regard the Pai as
the goal of the next planning
Put ai into the set list.
}
Computing the routing relations among the activities
in the set list and adding the routing relations into the
Δ;
}
Until (Available_goal is empty)
Return W;
}
```

AN EXAMPLE

To introduce the problems tackled in this research, consider a process of interaction between two business partners. Among the process is the contractor ordering products from the subcontractor at fist, then, the subcontractor manufacturing the desired product and sending it to the contractor. For simplicity, only the first step in this scenario, ordering is discussed. The ordering is typically used to describe an abstract view rather than a detailed ordering procedure or subtle aspects. That means the ordering can be considered as an abstract activity.

The abstract order activity can be separated as some sub execution activities: activity sending an order, activity sending a specification, activity creating cost statement, processing cost statement and signing a contract. To illustrate the automated reasoning process, the definitions of the corresponding semantic activities are shown as follows:

a_1 = send_order = (P_{a1}, E_{a1})
P_{a1} = {contractor (a), subcontractor (b), requested (a, b)}
E_{a1} = {order_infor(c), hold (b, c)}
a_2 = send_specification = (P_{a2}, E_{a2})
P_{a2} = {contractor (a), subcontractor (b), requested (a, b)} E_{a2} = {specification_infor (d), hold (b,d)}
a_3 = Create_cost = (P_{a3}, E_{a3})
P_{a3} = {order_infor(c), specification_infor (d), hold (b, c), hold (b, d)} E_{a3} = {cost_statement (f), subcontractor_confirmed (b), hold (a, f)}
a_4 = Process_cost = (P_{a4}, E_{a4})
P_{a4} = {cost_statement(f), hold(a, f),} E_{a4} = {contractor _confirmed (a)}
a_5 = Sign_contract = (P_{a5}, E_{a5})
P_{a5} = {contractor_ confirmed (a), subcontractor_ confirmed (b)}
E_{a5} = {contract_signed (a, b)}

The concepts used in the semantic activities above-mentioned are defined in the following acyclic TBox T:

T = {infor = order_infor∪specification_infor, contract_ confirmed = contractor _confirmed∪subcontractor_ confirmed}

The algorithm fist extracts the goal g of the abstract ordering activity, g = {contract_signed(a, b)}. Then, using reason mechanisms, the algorithm starts to search for the activity which can accomplish g, which the result is a_5.

Therefore, P_{a5} turns into the goal g_1 in the next planning, g_1 = {contractor_confirmed(a), subcontractor_ confirmed(b)}. Similarly, the activities achieving g_1 are a_3 and a_4. According to the deciding methods for routing relations, the routing relation among a_3, a_4 and a_5 is And-join. Add the a_4, a_5 into the list and ((a_3, a_5), (a_4, a_5)) into Δ. Through recursive computation, other activities associated with the current instance as well as the routing relations among them can be determined. However, in the next planning, the P_{a6}, cost_statement(f), will become the goal of the planning. Suppose that the planner finds three cost sheets from manufacturer1, manufacture r2 and manufacturer 3. However, the contractor has a constraint that he or she wishes to purchase an order from manufacturer 3. As a result, only the activity binding manufacturer 3 will satisfy the contractor's constraint and others will violate it. Thus, the activity binding manufacturer 3 will become the activity best suitable to the current instance process.

CONCLUSION

Today's workflow management systems are weak in dealing with the process changes. In the study, the research is focus on description logics based workflow ontology model and reasoning over the semantic activity and uses' preferences to deal with the problem, which provides the support for the changes of the process. However, the expression of the preconditions and effects in semantic activities is limited to the basic assertions. The extension of the formalization of semantic activities with number restriction constructors will be further worked on to argument the reasoning functions of the workflow ontology. Correspondingly, the deciding method for iteration routing will be further discussed.

ACKNOWLEDGMENTS

This research was supported in part by the National Natural Science Foundation of China under Grants 60773034, 90818023, 90718012 and 60803032; the Scientific and Technological Developing Program of Shandong Province of China under Grant 2008GG30001024; the Taishan Scholar Construction Project of Shandong Province, China; the Open Project of the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences under Grant SYSKF0804. The postgraduate students innovation fund of Shandong University of Science and Technology under Grant YCA090427.

REFERENCES

Baader, C., M. Lutz, U. Milicic, U. Sattler, F. Wolter, 2005. A description logic based approach to reasoning about Web service. Proceedings of the Workshop on Web Service Semantics, May 10-14, Chiba City, Japan, pp: 1-12.

Baader, F., D. Calvanese, D. McGuinness, D. Nardi and P.F. Patel-Schneider, 2007. The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, UK., ISBN-13: 9780521876254.

Gerevini, A. and D. Long, 2005. Plan constraints and preferences for PDDL3. Technical report, Department of Electronics for Automation, University of Brescia, Italy. http://www.cs.yale.edu/homes/dvm/papers/pddl-ipc5.pdf.

Han, L.Q., Z. Yao, S.F. Liu and Z.Q. Zheng, 2007. Application and Research of Ontology in Workflow Knowledge Meta-model. Proceeding of the 11th International Conference on Computer Supported Cooperative Work. April. 26-28, Melbourne, Australia, pp: 675-680.

Heinl, P., S. Horn, S. Jablonski, J. Neeb, K. Stein and M. Teschke, 1999. A comprehensive approach to flexibility in workflow management systems. Proceedings of the International Joint Conference on Work Activities Coordination and Collaboration. Feb. 22-25, San Francisco, California, United States, pp: 79-89.

Korhonen, J., L. Pajunen and J. Puustjhi, 2003. Automatic composition of Web Services Workflows using a semantic agent. Proceedings of the IEEE/MIC International Conference on Web Intelligence (WI'03). Oct. 13-16, Halifax, Canada, pp: 566-569.

Rinderle, S., M. Reichert and P. Dadam, 2004. Correctness criteria for dynamic changes in workflow systems: A survey. Data Knowledge Engineering, 50: 9-34.

Sadiq, S., W. Sadiq and M. Orlowska, 2001. Pockets of flexibility in workflow specifications. Proceedings of the 20th International Conference on Conceptual Modeling. Nov. 27-30, Yokohama, pp: 513-526.

Sirin, E., 2006. Combining description logic reasoning with AI planning for composition of web services. Ph.D. Thesis, Department of Computer Science, University of Maryland.

Son, T. and E. Pontelli, 2004. Planning with preferences using logic programming. Proceedings of the 7th International Conference on Logic Programming and No Monotonic Reasoning (LPNMR04). Jan. 6-8, Berlin, Heidelberg, New York, pp: 247-260.

Van der Aalst, W.M.P. and Kees van Hee, 2002. Workflow Management: Models, Methods and Systems. MIT Press, Cambridge, Massachusetts, ISPN: 0-262-01189-1.

Vieira, T., M. Casanova and L. Ferrb, 2004. An ontology driven Architecture for Flexible Workflow Execution. Proceedings of the Web Media and LA-Web 2004 Joint Conference 10th Brazilian Symposium on Multimedia and the Web 2nd Latin American Web Congress. 2004, Brazilian, Latin American, pp: 70-77.

Wang, S., W. Shen and Q. Hao, 2005. Agent Based Workflow Ontology for Dynamic Business Process Composition. Proceedings of the 9th International Conference on Computer Supported Cooperative Work in Design Proceedings. May. 24-26 Coventry University, UK, pp: 452-457.

Wu, Y.M., J. Lu, S.W. Yao and Z.J. Zheng, 2005. Mechanism of Semantic Oriented Flexible Workflow. Proceedings of the Third International Conference on Information Technology and Applications. Jul. 4-7, University Technology, Sydney, pp: 209-214.