Towards Service Coalitions: Coordinating the Commitments in a Workflow

Jiangbo Dang and Michael N. Huhns

Department of Computer Science & Engr., University of South Carolina
Columbia, SC 29208, USA. 1-803-777-3768
{dangj,huhns}@engr.sc.edu

Abstract. Web services are functionalities that can be engaged over the Internet. A workflow is a set of Web services that are executed by carrying out specified control and data flows among these services to address some business needs. We believe that commitments among agents can be used to model a workflow and coordinate several self-interested parties to execute a workflow. This paper presents a methodology to infer commitments and causal relationships from a workflow by utilizing semantic descriptions of Web services. We provide an example scenario to show how commitments of a workflow can be inferred. In addition, we use the Petri net representation of a workflow to describe our algorithm. With this technology, agents (service requestors and providers) engaged in a workflow can negotiate with multiple agents to reach favorable agreements and then coordinate their behaviors through the commitment operations in the context of service-oriented environment, where one or more self-interested parties can provide services to one or more other parties.

1 Introduction

In supply chains, e-commerce, and Web services, the participants negotiate contracts and enter into binding agreements with each other by agreeing on functional and quality metrics of the services they request and provide. The functionality of a service is the most important factor, especially for discovering services. Once discovered, however, services are engaged, composed, and executed by the participants' negotiating over issues besides QoS (quality of service) metrics to maximize their profits.

As more complex business operations become candidates for automation, it is difficult to find one Web service to fulfill a complete business process. The situation becomes even more complicated when there is no single Web service, but only a combination of several Web services that can satisfy a business need. To solve this problem, various standards for coordinating the Web services have been developed, such as BPEL4WS, OWL-S, and WS-Coordination. We use OWL-S in our paper since it provides richer semantic descriptions than the alternatives.

Semantic Web service technologies, such as OWL-S enable more flexible automation of service discovery and execution and monitoring, and support the composition of more complicated workflows represented as composite services. OWL-S provides a standard language for describing the composition of Web services. Thus we can treat composite services as a process model.

Negotiation is a process by which agents communicate and compromise to reach agreement on matters of mutual interest while maximizing their individual utilities. In a service-oriented environment, it is very likely there are multiple service requestors and providers negotiating simultaneously. Concurrent negotiation is preferred, since it is both time efficient and robust when an agent needs to negotiate with multiple other agents to make a good deal or to request a service involving multiple agents as in a workflow model.

Commitment among agents can be used to model business processes by capturing the interactions among agents. Chopra and Singh [2] proposed a commitment-based formalism to represent multiagent interaction protocols. To be coordinated with other agents in a workflow execution, all participating agents negotiate to reach beneficial contracts and coordinate their commitments to fulfill a business process. The commitments of the agents lend coherence to their interactions over time. For our work, we are interested in inferring commitments and causalities from a business process model and support multiple-issue concurrent negotiation for a workflow among collaborative parties in the future.

2 Background and Related Work

2.1 BPEL4WS, OWL-S, and SWRL

When we choose a representation for Web services, the trade-offs must be made among the expressive power, the rigor, the ease of use, and the computational tractability of a representation [10]. IBM, Microsoft and BEA released BPEL4WS (Business Process Execution Language for Web services) for expressing workflows consisting of Web services. BPEL4WS enables the specification of executable business processes (including Web services) and business process protocols in terms of their execution logic or control flow.

OWL-S is an initiative of the Semantic Web community to facilitate automatic discovery, invocation, composition, interoperation, and monitoring of Web services through their semantic description [4]. OWL-S supports a richer semantic description of Web services by: (1) a profile that describes what the service does, (2) a process model that specifies how the service works in terms of inputs, outputs, preconditions, and result, a.k.a., IOPR, and (3) a grounding that defines how the service is accessed. Both OWL-S and BPEL4WS provide a mechanism for describing a business process model. OWL-S augments the input and output specifications of BPEL4WS with preconditions and results: this enables the side effects of services to be encoded. We can then reason about how services may be composed and infer the commitments and causalities from them.

Semantic Web Rule Language (SWRL) expressions may be used in OWL-S preconditions, process control conditions (such as If-then-Else), and results.

SWRL expressions may also mention process inputs and outputs as variables, thus tying together the two languages. SWRL can make OWL-S more powerful, since it uses the expressive power of rules in a potential emerging standard. In our work, we just use SWRL in a primitive way, as a conjunction of conditions.

2.2 Web Services and Workflow

Semantic Web services, as envisioned by Berners-Lee, are intended to be applied not statically by developers, but dynamically by the services themselves through automatic and autonomous selection, composition, and execution. Web services are standard-based software components that can be accessed over the Internet by other software components [6]. Web services can vary in functionality from simple operations, such as a retrieval of a stock quote, to complex business operations, such as supply chain problems.

Many efforts have been made to automate service composition. In [6], the authors introduced a workflow composer agent to compose Web service workflows by finding and matching the semantic descriptions of Web services. Mandell and McIlraith [7] used a bottom-up approach to integrate Semantic Web technology into automating the dynamic discovery and binding of Web services. Chung et al. [3] presented a Web service framework to support collaborative product commerce. Given a workflow, what issues different participating agents negotiate to reach service agreements and how to coordinate agents to execute a workflow are still challenging problems. In this paper, we focus on how to infer commitments from a workflow and extract issues for collaborative service negotiation.

Given a workflow consisting of several services, service agents negotiate with one another and with resource agents to ensure that global constraints are not violated and that global efficiencies can be achieved. As described in [10], the service agents must be able to engage in negotiation, and they must be describable declaratively, not procedurally, in terms of high level abstractions. As a binary relationship binding two participants, a commitment is a proper abstraction for coordinating different parties of a workflow.

2.3 Negotiation and Commitment

Current standards for Web services do not support multiple-issue negotiations. As a result, several researchers have attempted to merge negotiation from the MAS domain into Web service selection and composition. Petrone [8] proposed a conversation model to enrich the communication and coordination capabilities of Web services by adapting agent-based concepts to the communications among Web services and users.

Multi-linked negotiation describes a situation where one agent needs to negotiate with many other agents about different issues, and the negotiation over one issue influences the negotiation over the other issues. Multi-linked negotiation becomes important in a workflow scenario where a service requestor negotiates with several service providers to reach agreements over a composite service.

Zhang et al. [14] presented a mechanism for multi-linked negotiation of task allocation in a cooperative system where a contractee tries to ask another agent to fulfill one of its subtasks that it cannot do itself. Since their protocol does not support concurrent negotiation, it is not possible for a contractee to coordinate among multiple subtasks.

In [13], an approach is proposed to deal with multi-linked negotiation in the context of task allocation. a partial order scheduler is used to find the consistence range for issues in each task and the relationships among them by sorting all issues with their flexibilities and dependencies. In their model, negotiation is viewed as a multidimensional search over multiple issues (time, cost, and the flexibility of the commitment). It is a plan-globally-then-negotiate-separately procedure in which there is no mutual influence among negotiation threads.

Commitments are a key element of the semantics of agent communications [9]. Commitment among agents can be used to model business processes by capturing the interactions among agents. In [11], researchers extract commitments from a set of conversions via Dooley graphs and map Dooley graphs to π-calculus. The formalization of π-calculus helps to derive useful properties and prove soundness of their models. To further apply commitment into Web service, they [12] integrate commitments into service specification by which service providers and requestors exchange commitments instead of messages. Most existing workflow technologies can only apply centralized methods to coordinate and monitor the execution of a workflow through the procedural specifications. In contrast, this paper advances the state of the art in the following ways: Our methodology (1) infers the commitments of service agents involved in a workflow; (2) allows flexible workflow coordination through commitments; (3) makes it possible for service agents to negotiate over issues from the inferred commitments to improve their utilities while optimizing the workflow (4) potentially provides a way to build a flexible and robust workflow by concurrent service negotiation.

The remainder of the paper is organized as follows: Section 3 introduces a motivating workflow scenario and provides its Petri net representation. Section 4 identifies control constructs of a workflow in different representations and describes the algorithm to derive commitments from a workflow. Section 5 discusses further issues related to negotiating a workflow, and Section 6 concludes.

3 A Motivating Scenario

In order to illustrate our methodology, we present a motivating workflow scenario where several parties work together to produce a product. In Figure 1, there are five service agents: *ProductRequestor*, *ProductMaker*, *Analyzer*, *PartsMaker*, and *Driller*. *ProductRequestor* agent A initiates this workflow by sending a product requirement to *ProductMaker* agent B. To meet A's requirement, B designs this product and send its design to the third party *Analyzer* C. C performs some specific tests to ensure this design will meet the requirements. Once the product design is approved, B will generate the requirements for different parts of this product and send them to *PartsMaker* agent D. *PartsMaker* D will design these

parts and send the design to C. If C approves the parts design, D will produce the parts for the product. In addition, if the design requires a specific treatment like drilling, a *Driller* agent E will drill the parts. Finally, *ProductMaker* B will polish the parts and assemble the product to finish this workflow.

This workflow is complicated because:

- From the workflow's view, the structure of the workflow is dynamic and uncertain, since it depends on the outputs/results from the antecedent processes. For example, the execution of process *DrillParts* depends on the output from the process *DesignParts*; therefore, we can not know in advance whether *Driller* agent E will be involved.
- From the participant's view, the processes or tasks it needs to perform are also uncertain. It depends on the input of *ReceiveAnalysisRequest* whether the *Analyzer* agent C performs *AnalyzeProductDesign* process or *AnalyzePartDesign* process. Moreover, C has to repeat its processes many times if the outputs from *DesignProduct* or *DesignParts* cannot pass the tests.

Due to the dynamic property of workflow, we believe that commitments and conditional commitments are the proper abstraction to characterize and coordinate collaborative service agents in a workflow.

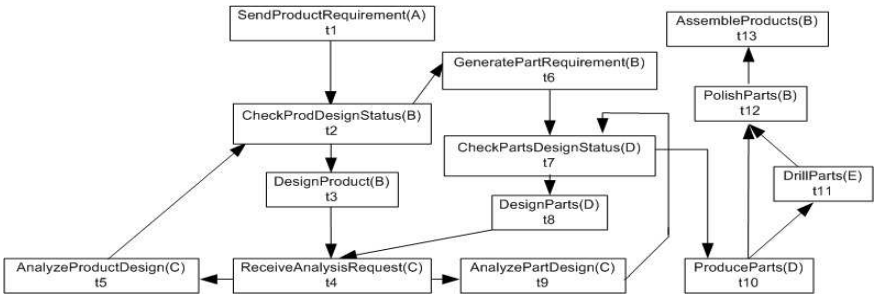

Fig. 1. A *ProduceProduct* Workflow Example

We describe the above workflow as a composite process in an OWL-S file. Its behavior is described in terms of its process model, where the functionality of each subprocess is described by its IOPR. OWL-S adopts two views of processes. First, a process produces a data transformation from a set of inputs to a set of outputs. Second, a process produces a transition in the world from one state to another. This transition is described by the preconditions and results of the process [4]. Inputs and outputs specify the data transformation produced by the process. Inputs specify the information that the process requires for its execution. The inputs are either provided by other processes in the process model or by service clients through message passing. Equivalently, the outputs are either sent to other processes through the data-flow constructs, or to other Web services. The execution of a process may also result in changes of the state of the world.

Preconditions specify conditions that should be satisfied for a process to execute correctly. The IOPRs for the *ProduceProduct* example are shown in Table 1.

Table 1. IOPRs of the Processes from the *ProduceProduct* Example

Process	Inputs	Outputs	Preconditions	Results
SendProdRequirement		ProductRequirements		
CheckProdDesignStatus	ProductRequirements AnalysisReport	ProductRequirements		Set Approved
DesignProduct	ProductRequirements	ProductDesign		
GeneratePartsRequirement	ProductDesign	PartRequirements		
PolishParts	Parts	Parts		Polished=true
AssembleProduct	Parts	Products		
ReceiveAnalysisRequest	Design DesignType	Design		
AnalyzeProductDesign	ProductDesign	?Approved AnalysisReport		
AnalyzePartsDesign	PartDesign	?Approved AnalysisReport		
CheckPartDesignStatus	PartsRequirements AnalysisReport	ProductRequirements		Set Approved
DesignParts	PartRequirements	PartDesign		
ProduceParts	PartDesign	Parts ?needDrilled		
DrillParts	Parts	Parts		Drilled=true

Conditions have a pervasive presence in OWL-S. They are used to describe outputs and results that result from the execution of processes. They are also used in the specification of constructs such as if-statements and loops. We use the primitive SWRL rules encoded as XML Literals. These SWRL rules uses rdf:List to represent a conjunction of expressions of true or false values.

In this example, the production of a product would follow the sequential process of receiving a requirement, designing a product or parts, and analyzing product design or parts design, producing a part, drilling if necessary, and polishing and assembling the product. *ReceiveAnalysisRequest* would involve the atomic process of either *AnalyzeProductDesign* or *AnalyzePartsDesign*. Moreover, designing and analyzing would be an iterative process.

Petri nets have been used to model and analyze many kinds of processes, and the colored Petri net extension facilitates the modeling of complex processes where data and time are important factors. Petri nets for workflow modeling provide: (1) a clear and precise formal representation, (2) an intuitive graphical language, (3) full expressiveness with explicitly represented states, and (4) a firm mathematical foundation for property investigation and analysis [1]. To illustrate our algorithm, we transform our example workflow into a colored Petri net. A Petri net $N = (P, T, F)$ consists of a set of transitions T (bars), a set of places P (ellipses), and a flow relation F (arcs) [1]. In a workflow Petri net, a transition represents an atomic process and a place is a passive state. Petri nets are well suited for modeling workflow processes, since there are many available simulation tools for them [1]. Therefore, we can test the Petri nets to determine the soundness and equivalence of workflows and those commitments inferred from a workflow.

A Petri net extended with color, time, and hierarchy is called a high-level Petri net [1]. In this paper we use the first extension to model conditions and relations of processes within a workflow. Other extensions are useful in dealing with time and scale issues of processes, which are beyond the scope of this paper. In a colored Petri net, each token has a value often referred to as 'color'.

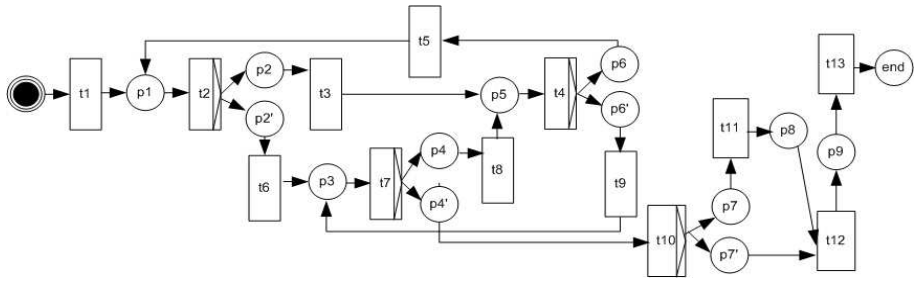

Fig. 2. A *ProduceProduct* Petri Net

Transitions determine the values of the produced tokens on the basis of the values of the consumed tokens, i.e., a transition describes the relation between the values of the 'input tokens' and the values of the 'output tokens'. It is also possible to specify 'preconditions', which take the colors of tokens to be consumed into account. These values match the inputs of a process, the outputs and results of a process, and the preconditions of a process from an OWL-S definition, respectively. Figure 2 shows the Petri net model of our example workflow. The details are discussed in Section 4.

4 Deriving Commitments from a Workflow

4.1 Workflow Control Constructs

Given a workflow, four types of routing are identified by the Workflow Management Coalition (WfMC) in specifying how cases are routed along the processes that need to be executed: sequential, parallel, conditional and iteration. In the process dimension, building blocks such as the AND-split, AND-join, OR-split, OR-join, explicit OR-split, and explicit OR-join are used to model the routing [1].

Sequential routing is used to deal with causal relationships between tasks. Consider t_1 and t_2 from Figure 2. If t_2 is executed after the completion of t_1, then t_1 and t_2 are executed sequentially. Place p_1 represents a result for t_1 and a precondition for t_2. Parallel routing is used in situations where two processes need to be executed, but the order of execution is arbitrary. Considering the two sets of *ProductMaker* and *PartsMaker* in our example, processes after t_1 can be executed in parallel. To model such a parallel routing, two building blocks are used: (1) the AND-split and (2) the AND-join. Conditional routing is used to

allow for a routing that may vary between cases. To model a choice between two or more alternatives, the explicit OR-split is used. In Figure 2, t_2 has two output places p_2 and p_2'. The choice between p_2 and p_2' is based on the attribute *Approved*. If *Approved* is true, t_6 will be executed, otherwise t_3 is executed. The iteration routing can be modeled using an explicit OR-split as the iteration of $t_2 - t_3 - t_4 - t_5$ defined in Figure 2. t_2 is a control task that checks the result of t_5. Based on this check, t_3, t_4, t_5 may be executed once more.

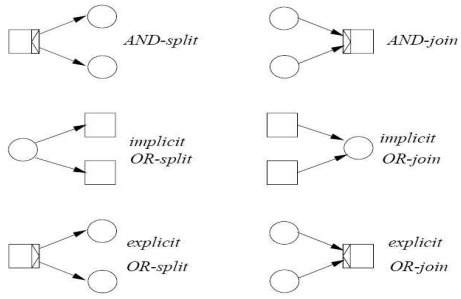

Fig. 3. The Building Blocks for Workflow Modeling [1]

All OWL-S control constructs can be categorized into the discussed four classes of routings or modeled by PN (Petri net) building blocks from Figure 3, e.g., OWL-S Sequence is equal to the PN sequential routing, OWL-S Split and Split-and-Join can be represented by the PN AND-split and AND-join, OWL-S choice is equivalent to the PN explicit OR-split, OWL-S Any-Order can be modeled by the PN parallel routing, OWL-S Condition and If-Then-Else constructs can be represented by the PN conditional routings, OWL-S Iterate, Repeat-While, and Repeat-Until may be modeled by the PN iteration routings.

4.2 Inference Algorithm

In service-oriented environments, the participating agents are distinguished by the services they provide, the services they seek, and the negotiated service agreements to which they commit. The coherent behavior of systems in such an environment is governed by interactions among the agents, and commitments are the proper abstraction to characterize the interactions for monitoring and control of the systems [5].

A service is what an agent performs when it works on and completes a task or process. In this paper, a workflow is represented as a composite service in OWL-S format where each sub-service is described by its IOPR properties. Each sub-service is associated with agents via a process of negotiation. The execution of the workflow is monitored via commitments. A commitment is a well-defined data structure with algebra of operations that have a formal semantics. The agent

that is bound to fulfill the commitment is called the debtor of the commitment. The agent that is the beneficiary of the commitment is called the creditor. A commitment has the form $C(a; b; q)$, where a is its creditor, b is its debtor, and q is the condition the debtor will bring about. A conditional commitment $C(a; b; p \rightarrow q)$ denotes that if a condition p is brought about, then the commitment $C(a; b; q)$ will hold. Commitments capture the dependencies among the agents with regard to the workflow and can be inferred by the algorithm 1.

We assume that the data flows and message mappings are well defined in the semantic description. Let $e(v_1, v_2)$ denote an arc from vertex v_1 to vertex v_2. Given a workflow defined as a Petri net $N = (P, T, F)$, we define a directed graph $N'(V, E)$ where $V = T$ and $e(v_1, v_2) \in E$ if $\exists p$, $e(v_1, p) \in F$ and $e(p, v_2) \in F$. The neighbor nodes of $v \in V$ are stored in $adjacent(v)$ and the color of each vertex $v \in V$ is stored in the variable $color(v)$. We define the start transaction v_0 as the root node of N'.

Since same service agent may execute several atomic processes in one workflow, we need to distinguish between the concepts of agent and role. A role is an abstraction of capabilities used by an agent in dealing with one atomic process. An agent may have several roles, each associated with one commitment. Algorithm 1 produces a set of commitments for service agents. Each commitment is represented as the OWL-S IOPRs, which can be easily transformed into the commitment format we defined in the previous section. These commitments can be used in two ways: coordinating and guiding the negotiations among service agents in a competitive service-oriented environment, and monitoring and controlling the debtor agents to fulfill the workflow by fulfilling their committed tasks. To make this possible, the services have to be defined with a semantic description, and the preconditions/results and inputs/outputs should refer to an ontology. Given IOPRs of the processes defined in Table 1 and the Petri net in Figure 2, let us illustrate Algorithm 1 with our example scenario. For *Driller E* with one process: *DrillParts*.

[DrillParts]
Input: Parts
Output: Parts
Precondition: Completed(ProduceParts) $\wedge$ needDrilled
Result: Drilled

For *ProductMaker B* that owns three atomic processes: *CheckProdDesignStatus*, *DesignProduct*, and *GeneratePartRequirement*.

[CheckProdDesignStatus]
Input: ProductRequirements $\wedge$ AnalysisReport
Output: ProductRequirements $\wedge$ AnalysisReport
Pre-conditions: Completed(SendProdReuqirement)
Result: Set Approved true or false

[DesignProduct]
Input: ProductRequirements
Output: ProductDesign
Pre-conditions: Completed(CheckProdDesignStatus)$\wedge$ $\neg$approved
Result:

Notations:
$type(i)$ is the routing block type from vertex i;
$Owner(i)$ is the debtor of the process i;
Q is an empty first-in, first-out queue;
$enqueue(i, Q)$ adds element i into Q;
$dequeue(Q)$ removes and returns the first element from Q;
Initialization:
foreach $v \in V$ **do**
    $color(v) \leftarrow WHITE$
end
$enqueue(v_0, Q)$;
$color(v_0) \leftarrow BLACK$;
Inference:
while $Q \neq \phi$ **do**
    $i = dequeue(Q)$
    foreach $v \in adjacent(i)$ **do**
        if $color(v) = WHITE$ **then**
            $color(v) \leftarrow BLACK$;
            $enqueue(v, Q)$;
        end
    end
    switch $type(i)$ **do**
        case $Sequence$
            for j,*where* $e(i, j) \in E$ **do**
                $precondition(j) = precondition(j) \wedge result(i) \wedge completed(i)$;
            end
            break;
        case $AND - split$
            forall j,*where* $e(i, j) \in E$ **do**
                $precondition(j) = precondition(j) \wedge result(i) \wedge completed(i)$;
            end
            break;
        case $Explicit - OR - split$
            forall j,*where* $e(i, j) \in E$ **do**
                $precondition(j) =$
                $precondition(j) \wedge result(i) \wedge completed(i) \wedge OR - condition(i)$;
            end
            break;
    end
    forall j,*where* $e(i, j) \in E$ **do**
        if $e(i, j) \in E \wedge owner(i) \neq owner(j)$ **then**
            remove e
    end
end

Algorithm 1: Commitment Inference Algorithm

[GeneratePartRequirement]
Input: ProductDesign
Output: PartsRequirements
Pre-conditions: Completed(CheckProdDesignStatus)∧ approved
Result:

5 Negotiation and Commitments for Workflows

In service-oriented environments, the participating agents negotiate and commit to a service agreement about the execution and completion of a workflow. During the negotiation, the agents communicate and compromise to reach an agreement on matters of mutual interest while maximizing their utilities. The negotiated agreements can be encapsulated as commitment promises [5]. These inferred commitments and relations can be used for collaborative service negotiation. Moreover, we can identify the significant paths or processes and improve the robustness of a workflow by duplicating vital services through negotiations.

In a competitive service-oriented environment, explicit representation of commitments is the proper abstraction to coordinate participating agents in a workflow since: (1) It refers to interagent dependencies through the IOPRs of a task, thus allowing agents to recognize focus points in the revision process where coordination with other agents is needed; and focusing the distributed search this way benefits the efficiency of coordination; (2) An agent first tries to revise task timings that do not involve its commitments during the process of revising its local plan, this heuristic modularizes the revision as much as possible, making it more scalable [5]. Therefore, a centralized workflow execution engine is not necessary for coordinating, monitoring the execution of the workflows, and for verifying the output of the workflow.

ebXML addresses the broad problem of B2B interaction from a workflow perspective. ebXML uses Collaboration Protocol Profiles (CPP) to describe the business processes supported by Web services. A Collaborative Partner Agreement (CPA), an intersection of two CPPs, represents a technical agreement between two or more partners. A business process in ebXML is considered to be a set of business document exchanges between a set of Web services. OWL-S descriptions could be used within ebXML to describe the business processes of interacting Web services. The negotiations and commitments considered in this paper provide a potential representation, semantics, and methodology for establishing the CPA in ebXML.

6 Conclusions

This paper presents a methodology to infer commitments and relations from a workflow by utilizing semantic descriptions of Web services. With a motivating workflow scenario, we provide its semantic descriptions with IOPRs and a Petri net representation. We first identify the control constructs and then describe the algorithm to derive commitments from a workflow.

There are several possible directions for future work. First, this method can be applied to support the negotiation for a composed service with different service agents under constraints such as QoS and dependency issues. Second, we can further explore the power of semantic rule language to describe the relations within a workflow. Third, a process algebra, π-calculus, can be adopted to improve the flexibility of the current commitment model.

References

1. W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.
2. Amit K. Chopra and Munindar P. Singh. Nonmonotonic commitment machines. In *Proceedings of the International Workshop on Agent Communication Languages and Conversation Policies (ACL)*. Springer, 2003.
3. Moon-Jung Chung, Hong Suk Jung, Woongsup Kim, Ravi Goplannalan, and Hyun Kim. A framework for collaborative product commerce using web services. In *ICWS*, pages 52–60, 2004.
4. The OWL Service Coalition. OWL-S: Semantic Markup for Web Services.
5. Jiangbo Dang, Devendra Shrotri, and Michael N. Huhns. Distributed coordination of an agent society based on obligations and commitments to negotiated agreements. In Paul Scerri, editor, *Challenges in the Coordination of Large-Scale Multiagent Systems*. Springer Verlag, 2005.
6. Mikko Laukkanen and Heikki Helin. Composing workflows of semantic web services. In *Proceedings of the Workshop on Web-Services and Agent-based Engineering*, 2003.
7. Daniel J. Mandell and Sheila A. McIlraith. Adapting bpel4ws for the semantic web: The bottom-up approach to web service interoperation. In *International Semantic Web Conference*, pages 227–241, 2003.
8. G. Petrone. Managing flexible interaction with web services. In *Proc. Workshop on Web Services and Agent-based Engineering (WSABE 2003)*, pages 41–47, Melbourne, Australia, 2003.
9. Munindar P. Singh and Michael N. Huhns. Social abstractions for information agents. In Matthias Klusch, editor, *Intelligent Information Agents*. Kluwer Academic Publishers, 1999.
10. Munindar P. Singh and Michael N. Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. Wiley, London, UK, 2005.
11. Feng Wan and Munindar P. Singh. Mapping dooley graphs and commitment causality to the pi-calculus. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 412–419, Washington, DC, USA, 2004. IEEE Computer Society.
12. Feng Wan and Munindar P. Singh. Enabling persistent web services with commitments. In *Information Technology and Management (ITM)(In Press)*, 2005.
13. Xiaoqin Zhang, Victor Lesser, and Sherief Abdallah. Efficient Management of Multi-Linked Negotiation Based on a Formalized Model. *Autonomous Agents and Multi-Agent Systems*, 2004.
14. XiaoQin Zhang, Victor Lesser, and Rodion Podorozhny. Multi-Dimensional, Multi-Step Negotiation for Task Allocation in a Cooperative System. *Autonomous Agents and MultiAgent Systems*, 2003.