Workflow Coordination for Service-Oriented Multiagent Systems

Jiangbo Dang
Siemens Corporate Research
Princeton, NJ 08540, USA
jiangbo.dang@siemens.com

Jingshan Huang
University of South Carolina
Columbia, SC 29208, USA
huang27@sc.edu

Michael N. Huhns
University of South Carolina
Columbia, SC 29208, USA
huhns@sc.edu

ABSTRACT

From a multiagent viewpoint, a workflow is a dynamic set of tasks performed by a set of agents to reach a shared goal. We show herein that commitments among agents can be used to model a workflow and coordinate their execution of it. From a service-oriented computing viewpoint, a workflow can be represented as a set of services and a specification for the control and data flows among these services to address some business needs. As a formal declarative knowledge representation model, ontology is used as a basis for agent-based workflow execution and coordination. This paper presents methodologies to map an Ontology Web Language for Services (OWL-S) representation for a workflow to a CPN graph, a graphical and mathematical modeling tool for describing and analyzing information processing systems, and then infer commitments and causal relationships from the CPN graph. We provide an example scenario to describe our algorithms.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; F.3.2 [**Semantics of Programming Language**]: Process models

General Terms

Algorithms

Keywords

workflow validation, coordination, agent, commitment

1. INTRODUCTION

Workflows are becoming ubiquitous in Business Process Management applications. To execute a workflow in an open enterprise environment, the participants must negotiate and enter into binding agreements with each other by agreeing on functional and quality metrics of the services they request and provide. Therefore, Multiagent Systems (MAS) will be the workflow enactment mechanism of the future.

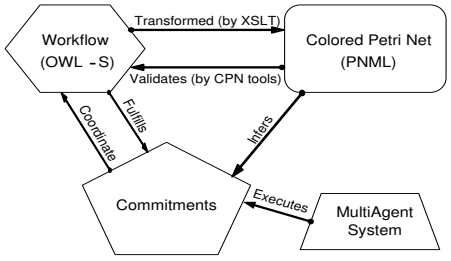

Figure 1: The relationships among a OWL-S workflow, a CPN, and commitments for service agents in an MAS implementation

An ontology serves as a declarative model for the knowledge and capabilities possessed by an agent or of interest to an agent. It forms the foundation upon which machine understandable service descriptions can be obtained, and as a result, it makes autonomic coordinations among agents possible. The use of and reference to ontology help the functionalities and behaviors of service agents to be described, advertised, discovered, composed, and coordinated in the execution of a workflow.

A commitment is a well-defined data structure with an algebra of operations that have a formal semantics. The agent that is bound to fulfilling 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 condition p is brought about, then the commitment $C(a; b; q)$ will hold. From a workflow viewpoint, q in $C(a; b; q)$ is the condition the debtor will bring about by fulfilling the task involved. Commitments among agents can be used to model business processes by capturing the interactions among agents.

As illustrated in Figure 1, we discuss the relationships among an OWL-S workflow, a PNML colored Petri net, and agents' commitments in a MAS, and present methodologies to infer commitments from a workflow. Most existing workflow technologies do not consider commitments, and apply only centralized methods to coordinate and monitor the execution of a workflow through its procedural specifications. In contrast, this paper advances the state of the art by describing how to (1) convert a semantic Web service model

AAMAS'07, May 14–18, 2007, Honolulu, Hawai'i, USA.

978-81-904262-7-5 (RPS)

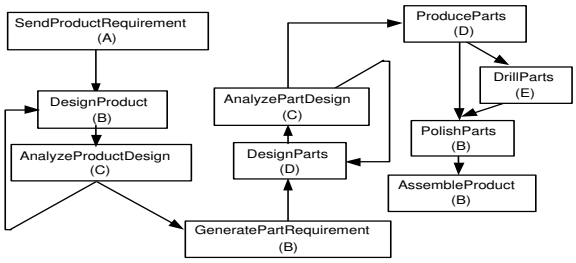

Figure 2: A ProduceProduct Workflow Example

(in OWL-S) to a graphic process model (in CPN) and infer the commitments of service agents involved in a workflow; (2) explore the use of colored Petri nets in workflow validation; (4) allow flexible MAS-based workflow coordination through agents' commitments.

2. A MOTIVATING SCENARIO

In a *ProduceProduct* workflow scenario, five parties work together to produce a product. In Figure 2, *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. 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, B will polish the parts and assemble the product to finish this workflow.

We first briefly present an ontology as a basis for workflow description and coordination in Figure 3, and then describe this workflow as a composite service, with its behavior described in terms of its inputs, outputs, preconditions, and results (IOPRs) in an OWL-S model. Current workflow coordination mechanisms cannot deal with this scenario properly due to its dynamic nature. We believe that commitments are an appropriate abstraction to characterize and coordinate collaborative service agents in a workflow.

3. INFERRING THE COMMITMENTS IN A WORKFLOW

A Petri net $N = (P, T, F)$ consists of a set of *transitions* T, a set of *places* P, and a flow relation $F(arcs)$. It is a directed, bipartite graph in which each node is either a *place* or a *transition*. *Place* is used to describe possible states of a process. The actions of a process are described by *transitions*. *Arcs* are used to connect places and transitions. P, T, and F are indicated by ellipses, rectangles, and directed lines in a PN diagram, respectively. There are *Tokens* in places. A transition is enabled if there is at least one token in every place connected to a transition, any enabled transition may *fire* by removing one token from every input place, and producing one token in each output place. In a workflow Petri net, a transition represents an atomic process, and a place is a passive state. Petri nets are well suited for

Figure 3: A ProduceProduct Ontology

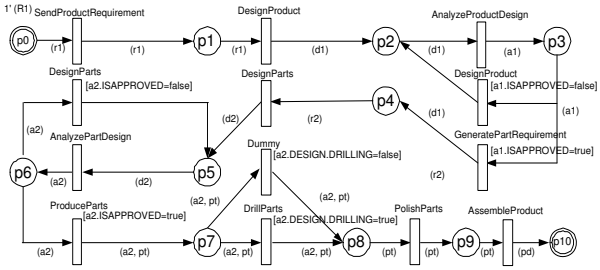

Figure 4: A ProduceProduct Petri Net

modeling workflow processes since there are many available simulation tools for them. Therefore, we can test the Petri nets to determine the validity and liveness of workflows and those commitments inferred from a workflow.

In an extension of a PN called a colored Petri net, each token has a value referred to as *color*, which can be a schema or type specification. transitions determine the values of the produced tokens on the basis of the values of the consumed tokens. It is also possible to specify a *guard* of a transition, which takes 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.

We can transform our example workflow from an OWL-S into a PNML CPN representation. The mapping algorithm is based on a depth-first search and yields valuable information about the structure of the workflow $T(V, E)$. V is a set of vertices and E is a set of edges. The neighbor nodes of $v \in V$ are stored in $adjacent(v)$, the color of each vertex $v \in V$ is stored in the variable $color(v)$, and the predecessor of $v \in V$ is stored in the variable $\pi(v)$. We perform the

Notations:
$type(i)$ is the node type of vertex i;
$parent(i)$ is the parent node of the vertex i;
t is the time;
Mapping(T):
foreach $u \in V$ **do**
  $color(u) \leftarrow WHITE$;
  $\pi(u) \leftarrow NIL$
end
$t \leftarrow 0$;
foreach $u \in V$ **do**
  if $color(u) = WHITE$ **then**
    $Mapping - node(u)$
  end
end
Mapping-node(u):
$color(u) \leftarrow GRAY$;
$t \leftarrow t + 1$;
$Processing - node(u)$;
foreach $v \in adjacent(u)$ **do**
  if $color(v) = WHITE$ **then**
    $\pi(v) \leftarrow u$;
    $Mapping - node(v)$;
  end
end
$color(u) \leftarrow BLACK$;

Algorithm 1: Workflow Mapping Algorithm

algorithm starting with the root node of T.

Processing-node(u) will generate a transition object if u is an atomic task and $type(parent(u)) = Sequence$ and an ingoing arc that links from the place of its predecessor transition, an outgoing arc, and a place object that the outgoing arc links to. If u is the first task of the workflow, it also generates a place object that links to the ingoing arc. In addition, *Processing-node(u)* will generate auxiliary objects if some control constructs are involved. Therefore, The input, output, precondition, and result of the OWL-S process can be stored in the inscription of the ingoing arc, the inscription of the outgoing arc, the guard of the transition, and the inscription of the outgoing arc, respectively.

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 $\exists p \in P$, $e(v_1, p) \in F$ and $e(p, v_2) \in F$. From our mapping algorithm, we know that the output PNML is restricted so that each transition has exactly one input and one output. Therefore, it can be done by including the input and output arc of each transition T inside this transition. 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 $color(v)$. The start transition v_0 is the root node of N'.

Considering that each 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. Given a Petri net workflow as the input, Algorithm 2 produces a set of commitments for service agents involved in a workflow.

Commitment $C(a; b; q)$ can be represented in terms of the IOPRs of q. Therefore, we can rewrite it as $C(a; b; (IOPR)_q)$

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;
Initialization:
foreach $v \in V$ **do**
  $color(v) \leftarrow WHITE$
end
$enqueue(v_0)$;
$color(v_0) \leftarrow BLACK$;
Inference:
while $Q \neq \phi$ **do**
  $i = head(Q)$
  foreach $v \in adjacent(i)$ **do**
    if $color(v) = WHITE$ **then**
      $color(v) \leftarrow BLACK$;
      $enqueue(v)$;
    end
    $i = dequeue(Q)$;
    for j,*where* $e(i, j) \in E$ **do**
      $precondition(j) =$
      $precondition(j) \wedge result(i) \wedge completed(i)$;
    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
end

Algorithm 2: Commitment Inference Algorithm

and a conditional commitment $C(a; b; p \rightarrow q)$ as $C(a; b; (I' OP'R)_q)$, where $I'_q = I_q \wedge O_p$, $P'_q = P_q \wedge R_p$. For *PartsMaker* that owns two tasks, one of its commitment is described as the following:

[DesignParts]	
Input:	PartRequirements
Output:	PartDesign
Pre-conditions:	Completed(GeneratePartRequirement) ∧ ISAPPROVED=false
Result:	

Commitments are the proper abstraction to coordinate participating agents in a workflow since they refer to inter-agent dependencies through the IOPRs of a task. After deriving the commitments from a workflow, the participating agents involved in the workflow can be monitored and coordinated. These commitments can be used in two ways: (1) Coordinating the interactions among service agents in a competitive service-oriented environment, and (2) monitoring the debtor agents to fulfill the workflow by fulfilling their committed tasks.

4. CONCLUSIONS

This paper discusses the relationships among an OWL-S workflow, a PNML colored Petri net, and agents' commitments in a multiagent system and presents methodologies to infer commitments from a workflow. the CPN representation can be analyzed for validity, deadlocks, liveness, and other faults by a variety of CPN tools. More importantly, agents can collaboratively enact a workflow through commitment-based formalisms by ontologically reasoning about their states and actions and produce coordinated and verified workflow execution.