OWL-P: OWL for Protocol and Processes

Ashok U. Mallya Nirmit Desai Amit K. Chopra Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh NC 27695 USA
{aumallya,nvdesai,akchopra,singh}@ncsu.edu

ABSTRACT

We describe OWL-P (OWL for Processes and Protocols), a methodology and software tool for specifying and enacting interaction protocols among autonomous agents. We use the Web Ontology Language (OWL) and the Semantic Web Rule Language (SWRL) to specify interactions as rule-based commitment protocols, which are a departure from the traditional, rigid protocol specifications. Protocols in OWL-P allow flexibility during execution and honor agent autonomy because of their declarative nature and their use of commitments. OWL-P separates *public protocols* from an agents' *private policies*, thus allowing protocols to be reused across varying contexts. OWL-P is a design time tool for business process designers and is of interest to researchers in multiagent systems and software engineering.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial IntelligenceDistributed Artificial Intelligence[Multiagent Systems]

General Terms

Design

Keywords

Agent Interaction, Business Processes, Semantic Web, Software Engineering

1. INTRODUCTION

Multiagent systems are a natural fit in applications that involve large, open systems with several autonomous entities that interact. Business processes are one such area where businesses with different and sometimes competetive interests have to interact with each other. Whereas predefined interaction patterns, i.e., protocols, make for efficient interactions, the rigidity of traditional protocol models such as state machines and Petri nets stifles agent sutonomy, thus defeating the chief benefit of employing agent-based systems. We develop a novel system and methodology with the following main features:

1. Declarative protocol specification with an underlying model that is based on commitments among the participants to achieve flexibility in execution.
2. A clear separation between publicly specified protocols and private local policies of interacting agents.
3. The ability to compose processes by combining protocols using a protocol algebra [3], derive individual participants' roles from the processes, and bind participants' policies at runtime.

Overview. We call our framework OWL-P [5, 2]. We use OWL, which is an RDF based description framework, to develop an ontology of concepts about commitments, protocols and processes. OWL-P represents protocols as rule-based transition systems using SWRL and gives first-class status to the commitments of the participants. This representation maximizes the flexibility of the agents enacting a given protocol and enables them to handle exceptions that may arise during enactment. Protocols can be composed to yield more protocols, and can be instantiated with agent policies to yield business process specifications.

Scope and Significance. This project reflects conceptual and theoretical work that we have been carrying on for several years. OWL-P is the first demonstration of a tool based on the concepts of commitment protocols. It will interest researchers in agent communication, multiagent organizations and teamwork, agent-based software engineering, and agent-oriented information systems. The demonstration will show the following main steps: a brief walk through OWL-P concepts and architecture, how to specify protocols in OWL-P, how to secify desired compositions of protocols in OWL-P, how to introduce policies to capture local requirements, how to instantiate a business process in OWL-P, and how to enact a business process in the face of different exceptions.

2. COMPONENTS

This section briefly describes the components of OWL-P. Protocols in OWL-P are message based. As shown in Figure 1, protocols are specified in the public domain in OWL-P in terms of the following :

1. The types of messages and their parameters, senders, and receivers.

AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.

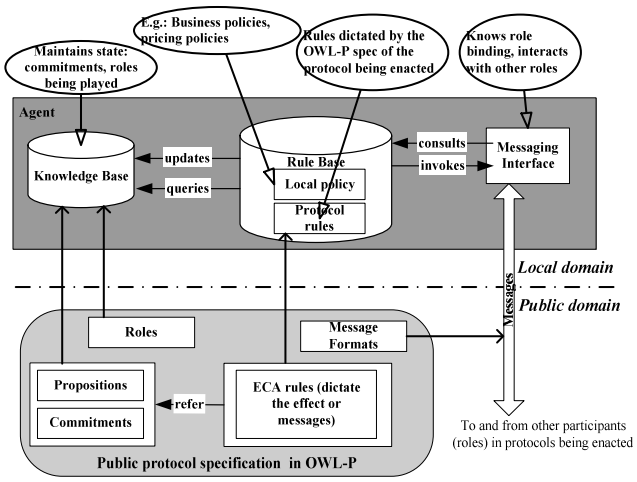

Figure 1: Features and components of an autonomous agent in the OWL-P framework

2. Propositions that model the universe of the protocol and the status of commitments.
3. Event-Condition-Action rules that specify the effects of a messge on propositions.
4. The roles that agents bind to while interacting using the protocol.

Since protocols are not specified as an explicit ordering of messages, agents send messages after consulting their local policies to determine when to send which mesage, for instance a book-seller agent's policy might determine how long to wait before shipping a book, how much to charge for a book, and so on. In this manner, OWL-P respects and promotes agent autonomy.

3. USAGE

We envision OWL-P being used by designers of business processes that span multiple, autonomous businesses. We define a process an interaction modeled by a composition of protocols and enacted after incorporating local policies of participants. This is shown in Figure 2. In this example, a software designer creates a specification of a *Purchase* process in OWL-P by composing three existing protocols, *Order*, *Shipping*, and *Payment*, which are accessed from a protocol repository (1). The designer specifies a set of *axioms* that dictate data-dependency and ordering constraints among these protocols (2). An example of a data-dependency axiom is that the amount to be paid by the Customer to the Merchant in the Payment protocol is the same as the price quoted for an item by the Merchant in the Order protocol. An OWL-P Composer creates a new protocol (3).

Once a new protocol is created, it is registered in the protocol repository (4). For enactment,a merchant that wants to use this protocol derives its skeleton (5), which is the part of the protocol that is visible to it, combines its policy (6), and publishes its process to a registry (7). Customers who search the registry (8) and find this merchant (9), query the merchant for the Purchase protocol and derive their skeletons from it (10), after which they combine their local policies with their skeleton (11).

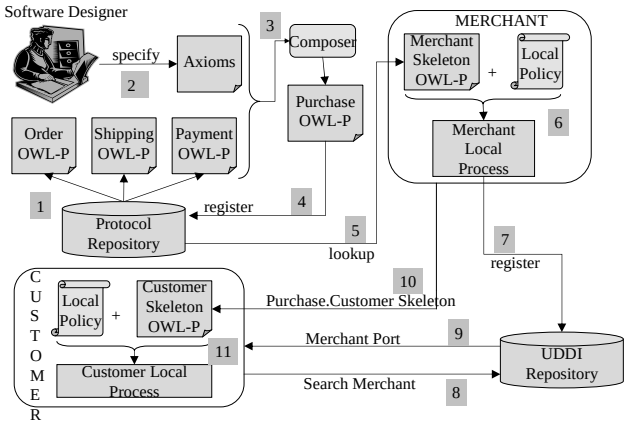

Figure 2: How OWL-P is used for processes design, composition, and enactment

4. CONTRIBUTIONS

The following are chief advantages of OWL-P.

- The specification and enactment of flexible protocols.
- A formal model for protocols which enables protocols to be modularly composed from existing ones.
- A clear separation of public behavior of participants of a protocol from their private policies.

Related Work. OWL for Services (OWL-S) [4] describes a semantic markup of services so that a planner working on behalf of an agent can compose services. OWL-S, however, takes one agents view, and therefore does not model multi-pary interactions. Since interactions are not modeled as separate entities, OWL-P does not permit their re-use as OWL-P does. The Business Process Execution Language (BPEL) [1] is has become the *de facto* standard for specifying and executing business processes. BPEL is a state-based process formalism and does not separate local policies from public protocol specifications. Hence BPEL severely limits component reuse. Also, BPEL does not take into account the autonomy of the participants of a process.

5. REFERENCES

[1] T. Andrews *et al.*Business process execution language for web services, (BPEL 1.1), May 2003. www-106.ibm.com/developerworks/webservices/library/ws-bpel.

[2] N. Desai, A. U. Mallya, A. K. Chopra, and M. P. Singh. Processes = Protocols + Policies: A methodology for business process development. Technical report, NC State University, 2004. TR2004-34.

[3] A. U. Mallya and M. P. Singh. A semantic approach for designing commitment protocols. In R. V. Eijk, editor, *Developments in Agent Communication*, volume 3396 of *LNAI*, pages 37–51. Springer, Berlin, 2004.

[4] D. Martin *et al.*OWL-S: Semantic markup for web services. W3C Member Submission, November 2004. www.w3.org/Submission/OWL-S.

[5] M. P. Singh, A. K. Chopra, N. Desai, and A. U. Mallya. Protocols for processes: Programming in the large for open systems. In *OOPSLA Companion*, pages 120–123, 2004.