Multiagent Policy Architecture for Virtual Business Organizations*

Yathiraj B. Udupi
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
ybudupi@ncsu.edu

Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
singh@ncsu.edu

Abstract

A virtual organization (VO) is a dynamic collection of entities (individuals, enterprises, and information resources) collaborating on some computational activity. VOs are an emerging means to model, enact, and manage large-scale service computations.

VOs consist of autonomous, heterogeneous members, often exhibiting complex behaviors. Thus VOs are a natural match for policy-based approaches. Traditional policy-based frameworks emphasize reactive behaviors, wherein an external request causes a policy engine to compute a response. However, business service settings require richer policies and call for proactive behaviors. A business not only must respond to explicit requests, but also monitor its environment, collate events, and potentially act in anticipation of events in order to ensure that its policies are satisfied. Autonomous, heterogeneous, proactive entities are best modeled as agents and, therefore, VOs are best understood as multiagent systems.

Our main contributions are (1) a proactive multiagent policy-based architecture, (2) a hierarchical model of policy monitoring, compliance checking, and enforcement for VOs, and (3) a formalization of VOs. We evaluate our approach using a real business service scenario.

1 Introduction

Virtual organizations (VOs) are dynamic collaborative collections of individuals, enterprises, and information resources [5]. Traditionally such collaborative activities are focused on data sharing and computation. This paper emphasizes VOs in business settings, especially where processes support delivery of real-world (not just IT) services. Because of legal and economic pressures, business environments provide richer policies than the more common scientific computing environments. VOs, whether business or scientific, have key properties that distinguish them from traditional IT architectures:

Autonomy. The members of a VO behave independently, constrained only by their contracts.

Heterogeneity. The members of a VO are independently designed and constructed, constrained only by the applicable interface descriptions.

Dynamism. The members of a VO join and leave with minimal constraints. Thus, the configuration of a VO changes at runtime.

Structure. VOs have complex internal structures, reflected in the relationships among their members.

Importantly, even in cases where the above properties are not required (such as within an enterprise where the members are controlled by one party), it is appropriate to architect a VO as if it had the above properties. Taking these properties to heart helps make the conceptual model neater and the designs more easily reusable and extensible.

The above properties of VOs closely match the properties of multiagent systems. Agents are persistent computations representing independent principals: they are *autonomous* and *heterogeneous* as a result. Multiagent systems are motivated from flexible human organizations and consequently exhibit *dynamism* and *structure*. Thus the distinguishing properties of VOs are mirrored in multiagent systems. As a result, we can leverage multiagent systems to model VOs, especially as applied in service computing [11, ch. 15–17].

The main contributions of this paper are (1) a proactive, multiagent policy-based architecture for VOs; (2) a hierarchical model of policy monitoring, compliance checking and enforcement for VOs; and, (3) a formalization of VOs. We compare our proposed policy architecture with traditional policy architectures using an example from a real

*This research was supported by the National Science Foundation under grant ITR-0081742.

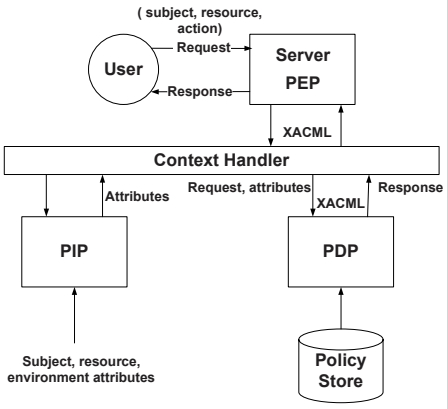

Figure 1. XACML policy framework

business scenario of VOs.

Organization. Section 2 describes the proactive policy management architecture for VOs containing a hierarchical organizational structure. Section 3 provides a formal representation for VOs. Section 4 describes a real business scenario of a VO used to evaluate our approach. Section 5 discusses the relevant literature. Section 6 summarizes our contributions and discusses future work.

2 Multiagent Policy Architecture

We briefly describe concepts on which our proposed approach builds. Section 5 describes other literature.

General policy architectures. The IETF Policy framework was designed for managing network resources [6; 15], but has a structure that applies more generally. This architecture assumes a network element (router, hub, or switch) where resource allocation decisions are made. A *policy enforcement point* or PEP resides within the network element. A *policy decision point* or PDP may be within or outside the element; policy decisions are made here. A PDP may use additional mechanisms to decide on an action.

XACML, the extensible access control markup language [9], extends the IETF framework. XACML has an access decision language used to represent a runtime request for a resource. First, a policy associated with a resource is located. Next, the attributes of the request are compared with the rules, ultimately yielding a permit or deny decision. XACML supports a runtime per-request access control mechanism and is illustrated in Figure 1. When a request is received, the recipient's policy enforcement point (PEP), forwards the request to a context handler along with some requester attributes. The context handler notifies the policy decision point (PDP). The PDP collects the required attributes from the policy information point (*PIP*) via the context handler and the appropriate policies from the policy store, and arrives at an authorization decision based on the attributes and the policies. The PDP returns the authorization decision back to the PEP, which enforces the returned policy decision on the client. XACML supports obligations; the PEP ensures that these obligations are met.

Grid policy architectures. Grid research has considered VOs and policy architectures. However, Grid policies focus on resource usage, access control, membership, and resource management. Dumitrescu et al. [3] support usage policy enforcement both at the VO level (for grid-wide policies), and at the site level (for local policies). Accordingly, their architecture includes two kinds of PEPs: S-PEP (site-level) and V-PEP (VO-level). An S-PEP ensures that only jobs satisfying local policies run at a site; others are preempted. A V-PEP schedules jobs on different sites. V-PEPs interact with S-PEPs and schedulers to enforce VO-level policies. Each PEP is supported by a monitoring distribution point (MDP), which gathers information about resource usage and policy restrictions. MDPs are distributed but can interact with other MDPs.

2.1 VO Policy Architecture Requirements and Proposed Approach

In a dynamic VO, the object of a policy need not be a passive entity but could also be an agent—usually one to whom the VO might have delegated some responsibility and granted some authority and visibility for the purposes of a specified family of business interactions. A desirable policy architecture would be flexible enough to capture dynamic relationships between entities in VOs, and the context in which the entities exist. Policies should kick in automatically leading to the creation of new VOs with appropriate assignment of roles and authorizations. We postulate the following *facets* of a policy, which are necessary for the practical policy management.

Creation. Policy scoping and creation corresponds to forming new VOs, and creating contracts between existing or new VOs.

Enactment. Policy enactment means acting in a manner to ensure compliance with the policies. Enactment involves delegation management in hierarchies, enforcement of policies, and so on.

Monitoring. Policy monitoring includes observing interactions among different parties and their outcomes.

Compliance. Policy compliance checking means determining at runtime whether a policy has been violated. It may include the anticipation of looming problems that would cause a policy to be violated, and thus may motivate evasive action.

The proposed approach goes beyond XACML in introducing additional architectural components geared toward monitoring and compliance. It borrows, from traditional grid policy architectures, the notions of monitoring and aggregating distributed events, and hierarchical enforcement of policies. However, we extend these notions in key respects, including (1) history-based event monitoring and gathering to support proactive mechanisms (2) compliance, (3) modeling and handling recursively formulated VOs, and (4) an architecture supporting autonomy and compliance.

2.2 Agent Representations

Our approach is centered on the notion of *agents*. An agent is a computational entity with a persistent identity that is proactive and interactive. As a base case, an agent may be an individual, such as a person, business partner, or resource. An agent may also be a VO. That is, a VO is an agent that comprises other agents, in particular, other VOs. We recently proposed an agent-based conceptual model for virtual organizations that emphasizes commitments and contracts [12].

This subsection describes the constructs of goals and policies that apply to agents, individuals or VOs. The next subsection introduces constructs that are specific to VOs. A member of a VO is referred to as its *child*, so that kinship terminology such as parent, descendant, ancestor, can be used.

Goals. The *goals* of an agent capture the states of the world that the agent desires to bring about. Goals are most naturally thought of as ends, but they can readily serve as means to other goals. In connection with VOs, the goals (ends) of some agents may cause them to enter into a contract or form a VO. Conversely, the contracts that a VO enters into may cause it to adopt goals (means), which could potentially yield additional goals for its members.

Policies. Each agent has its policies based on its business goals. An agent contains a PEP, a PDP, and a policy store, which function as in XACML. Figure 2 describes our agent architecture, which introduces two components, the *Policy Monitoring Point* (PMP) and the *Policy Compliance Checking Point* (PCCP), to support proactive behavior. The PMP monitors the agent's actions and its environment, and stores its observations in an event store. The PCCP monitors the event store. It can override the actions taken by the PEP, and can cause actions based on anticipated events. The events captured by an agent's PMP include the agent interactions and their outcomes, actions taken by the agent, dynamic agent relationships, and the current state of the entity represented by the agent.

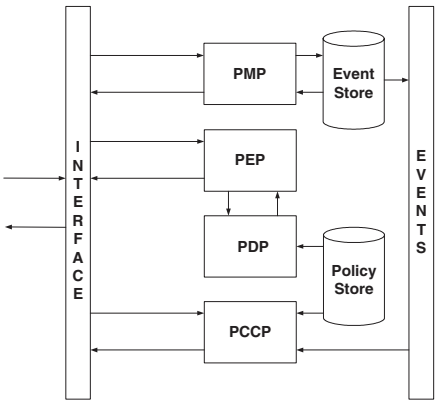

Figure 2. A proactive agent policy architecture

2.3 Organizational Representations

Intuitively, business partners may interact by sharing resources, requesting and providing services, delegating a task to one another, fulfilling a commitment, and forming contracts. Organizational representations formalize such interagent concepts.

Commitments and organizational context. The central primitive for expressing organizational interactions among agents is that of *commitments* [10]. Commitments function like directed obligations from a *debtor* to a *creditor*. Importantly, commitments are defined within an organizational *context*. Commitments specify a condition that the debtor is obliged to bring about. Conditional commitments associate this condition with a precondition. Such commitments provide a natural basis for contracts.

In essence, commitments reify aspects of agent interactions and enable interactions to be treated as first-class citizens in our representations. Six operations are defined on commitments [10]. Of these, the following are studied in this paper. A debtor of a commitment *discharges* it by bringing about the stated condition. A debtor of a commitment may *delegate* it to another agent: the outcome of the delegation is that the delegatee becomes the debtor of a commitment with the same condition and creditor as the original commitment.

The motivation for explicitly representing the context of a commitment is to delimit the scope of a commitment, so as to enable the proper treatment of exceptions and opportunities. In particular, commitments in real life are revocable: often, an agent has no choice but to revoke a commitment because of problems that may be, for instance, physical (factory burned down), economic (oil prices shot up unexpectedly), or legal (cannot ship pharmaceuticals across national boundaries). The context of a commitment provides a way to revoke or otherwise manipulate commitments. Delegation is an especially important variety of commitment

manipulation, and is discussed at length below.

Contracts. A *contract* among two or more agents encapsulates a related set of commitments. Typically, each of the parties to a contract would be the creditor of some commitment and the debtor of some commitment in the set. Also, typically, most of the commitments would be conditional and may refer to the conditions of other commitments in the set.

Contracting parties become members of the same VO, in essence. The act of contracting creates this VO. A contract describes how the participating VOs engage and collaborate with each other to deliver suitable services.

In the proposed approach, each commitment exists within the scope of a context VO. This leads to a coherence requirement for contracts: *Each of the commitments in the set that constitutes a single contract must have the same VO as their context.*

Delegation. The actions and interactions required for a contract may be carried out by members of the contracting VOs. The contracting VOs would delegate their commitments (to achieve certain goals in the contract) to their members. In business settings, delegation is routine. For example, say North Carolina State University (NCSU) contracts with the IDA Agency to have the COE building trimming painted. The contract specifies the stated service. NCSU delegates the tasks of scheduling, facilitating, and judging the paint job to its Facilities department. IDA delegates the job to its Raleigh division, which would deal with NCSU Facilities.

Each party to the contract acts in accordance with its policies. Because of delegation, lower organizations must adopt the contractual restrictions determined by higher organizations, which might potentially cause some of their local policies to be overridden. For example, Facilities may have a local policy that allows a building to be painted only on student holidays and IDA-Raleigh may have a policy to paint the outside only if the ambient temperature is below 80°. However, NCSU's policy of getting the building ready in time for school may necessitate overriding Facilities' policy and force painting on a day that is not a holiday.

Structure. A VO potentially incorporates complex relationships with its members and among its members. In our approach, these relationships are expressed in terms of goals, policies, and commitments.

For example, the goals of a VO can be propagated to its members as goals, or may become the commitments of its members. Likewise the policies of a VO would normally be propagated as policies of its members. The policies of a VO might control how the commitments among its members evolve. Consequently, as an important example, if two agents enter into a contract, besides the commitments that are explicitly part of the contract, their behavior would be constrained by the goals, policies, and commitments of their common parent VO. For instance, the parent VO might declare a contract invalid or successfully completed, or might release one of the agents from its commitments according to the contract. This level of flexibility is essential for a VO to handle exceptions and accommodate opportunities.

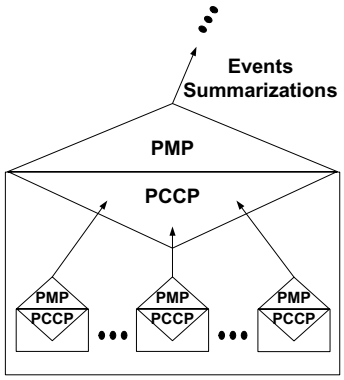

Figure 3. A hierarchical PMP–PCCP structure

Thus a VO can have policies, and an agent that is a member of a VO would normally be expected to satisfy any policies defined by the enclosing VO.

2.4 Hierarchical Enforcement

To facilitate management, organizations usually need to be hierarchical. The hierarchical structure of an organization, coupled with the distributed nature of its members, has interesting consequences on policy enforcement.

Figure 3 describes the hierarchical arrangement of PMPs and PCCPs of multiple agents belonging to an organizational hierarchy. Each VO is an envelope that potentially contains still more VOs as members. The PMP of a VO talks to the PCCPs of its parent VOs (more than one in case of multiple inheritance).

That is, each PMP summarizes events, and forwards these summarizations to its parents' PCCPs. The parent PCCP assimilates summarized event streams from multiple PMPs, and may initiate further action on the subordinate agents. Depending on its policies, a parent PCCP can override the functioning of its children: this follows from the computations specified above for an agent.

In the NCSU-IDA example, consider the situation when IDA-Raleigh takes longer to finish painting than expected. IDA-Raleigh's PMP observes this slow performance and notifies its PCCP. Now IDA-NC's PCCP may override IDA-Raleigh's policy, and make its staff work even when temperatures exceed 80°.

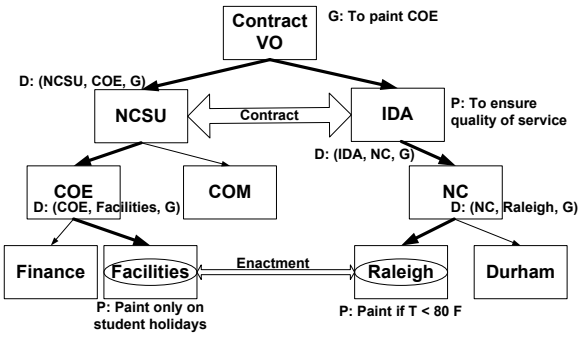

Figure 4. An example VO

3 Formal Representation

Based on the foregoing motivations, we formalize a VO as follows. Recall that a VO is an agent; an agent must be a VO or an individual.

Definition 1 A VO A is an agent defined via a tuple $\langle M, G, P, S, D\rangle$. Here $M = \{A_1, \ldots, A_n\}$ is a set of agents. G, P, S, and D are sets of formulas. The A_i are the members, G are the goals, and P are the policies of A. S is a set of commitments, each of which has a creditor and a debtor drawn from $M \cup \{A\}$ and a context equal to A. D is a set of delegations from A to A_i.

The goals of an agent are conditions (expressed as predicates applied to arguments), which the agent must make true, acting solo or collaboratively. The goals correspond to events, services, or tasks. Figure 4 represents a VO with two contracting members: NCSU and IDA. Only the non-trivial G, P, D components are shown. The commitments S are not shown, but correspond to the commitments of a VO to accomplish the goals specified.

Point of enactment. As described above, a goal is a condition to be achieved. For a well-formed VO, each goal must be reflected in the conditions of one or more commitments in a contract that exists within the scope of the VO. The debtor of this commitment may delegate it to one of its members. The delegation can continue down the VO structure. The point of enactment is the final debtor in the *delegation path* who directly discharges the commitment, and accomplishes the goal.

Definition 2 A point of enactment A_e of a goal e in a VO A, is a descendant of A that accomplishes e. That is, A_e is the debtor for a commitment S_e whose condition is e.

In Figure 4, the two rectangles enclosing ellipses referring to Facilities and Raleigh are the two points of enactment, belonging to the NCSU and IDA VOs, respectively.

3.1 Formal enactment of contracts in a VO

A contract may form in a top-down or bottom-up manner. Our approach represents both varieties uniformly via commitments. In a top-down setting, the contracting agents are already part of the same VO (i.e., are siblings), whose G component includes the goals of the contract. The contracting agents can delegate their commitments to any of their children. Figure 4 illustrates a contract enactment of this approach. A contract is formed between NCSU and IDA, both members of a common VO.

In a bottom-up setting, the points of enactment for a goal are responsible for the formation of a new common parent VO. A VO creation request can bubble up in the VO until it arrives at a suitably authorized party, which can enter into binding contracts. In both approaches, *delegation path* is defined as follows.

Delegation path. A delegation path begins from the VO within which a contract is formed, going down the VO hierarchy to the points of enactment, in essence propagating the commitments from the VO to the points of enactment.

Definition 3 A delegation path (q_e) of a goal e in a VO A is the path from the starting point of delegation (i.e., A), down the VO hierarchy to a point of enactment A_e.

In Figure 4, the paths marked with thick arrows starting at the Contract VO and ending at the points of enactment are delegation paths for the goal to paint COE. One path belongs to NCSU; the other to IDA.

Definition 4 Let Q_e be the set of all possible delegation paths for goal e of A_e. A path $q_e \in Q_e$ is *complete* if the successful discharge of the corresponding commitment (by the point of enactment in q_e), satisfies the policies of all the agents in the path.

In Figure 4, path q: $\langle$VO $\rightarrow$ IDA $\rightarrow$ NC $\rightarrow$ Raleigh$\rangle$ is complete with respect to the goal, if the goal is enacted in conditions with temperature lower than 80°, and if the desired quality of service is ensured. Path q is complete because the policies at all levels are satisfied. However, if a path q' to the Durham node existed, and if IDA-Durham had a policy restricting paint jobs to within Durham city limits, then q' would not be complete for the current goal, because NCSU's COE being located outside of Durham would cause a policy to be violated. The path from VO to Facilities via NCSU is also complete if the goal is enacted on a student holiday, and if the quality of service is ensured.

The enactment of a goal by a VO requires its members to have policy-compliant interactions to accomplish the goal. And, especially, if there is a hierarchical structure, then

the policies at all levels higher than the point of enactment should be satisfied.

The following theorems formally show the accomplishment of VO goals.

Theorem 1 A goal e is accomplished by a VO if each path in the set Q_e is complete.

This theorem follows from Definition 4, because the respective commitment for each path is discharged in accordance with the policies of all the agents in the path, thereby accomplishing the VO goal.

In our example, the goal to paint COE is accomplished by the VO containing the members NCSU, IDA, and others, provided both the paths are complete. This would be true and the theorem would apply if the goal were enacted on a student holiday, when the temperature was lower than 80°, and the quality of service was ensured. Because of a resulting incomplete path, the theorem would not apply if Durham were included in the coalition. Theorem 2 generalizes over Theorem 1 by considering all the goals in a VO.

Theorem 2 The goals G of a VO are accomplished if each goal $e \in G$ is accomplished in a manner that satisfies the policies P, commitments S, and delegations D formed by the VO and its members.

An interesting future direction is to study the scheduling of goals in a VO using heuristics such as earliest time of start and latest time of completion, and so on.

4 Evaluation: An Example Scenario

We consider a real business scenario of an insurance claim processing previously studied in the CrossFlow project [2]. We evaluate our approach with respect to the following three aspects: (1) an ability to model real business scenarios as hierarchical VOs, (2) proactive organizational enactment of policies locally in an entity, and hierarchically in a VO setup, (3) distributed event monitoring and compliance in a hierarchical setup.

Figure 5 illustrates the roles and operations involved in the claim processing. R_1 is AGFIL, the insurance company, who underwrites the policies and processes claims. R_2 is Europ Assist (EA), a 24-hr help-line service for receiving claims. R_3 is Lee CS (Lee), a consulting firm that handles claims. R_4 is Garage (Ga), a repairer. R_5 is the assessor (A), who assesses the repair cost estimates. R_6 is the insured (I), an insurance buyer. The roles of the insured and the assessor are not shown in Figure 5. Each role is represented by an agent that instantiates our proactive architecture.

In this scenario, we consider two VOs that are formed dynamically. These VOs may overlap with each other. The

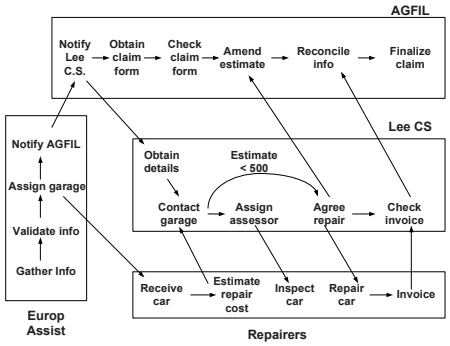

Figure 5. CrossFlow insurance claim scenario

nonempty components of these VOs, namely, *Goals (G)*, *Delegations (D)*, and *Policies (P)* are listed below.

VO-1 I and AGFIL.

G: The insured has to buy a policy from AGFIL, and AGFIL has to make the insured a policy holder.

P: The insured buys an insurance policy from AGFIL if AGFIL's quote is the best. AGFIL keeps the insured as a policy holder if the insured does not attempt fraud.

VO-2 AGFIL, Lee, A, and Ga.

G: AGFIL finalizes the claim by reconciling details such as repair estimates.

D: AGFIL delegates the commitments of claims monitoring to Lee. Lee delegates repair cost estimation to the garage Ga. Lee delegates repair cost estimate assessment to an assessor A.

P: Lee has a policy that an assessor be used if the repair estimate is greater than $500.

Proactive policy enforcement. Consider VO-1. The insured has a policy to select the insurance company that offers the best quote for the insurance policy. His PMP can keep track of the quotes provided by different insurance companies who contact him from time to time and store these as events in his event store. The PCCP monitors these events and can override the current policy enforcement by replacing AGFIL with another insurance company that gives a substantially better quote. AGFIL's PMP can keep track of the insured person's behavior in terms of the claims processed earlier. The PCCP monitors these events and can withdraw its policy if any fraudulent behavior is detected.

Definition 5 An agent A_i is *locally proactive (LP)* if it accomplishes its goals based on monitoring events and proactively enforcing its policies.

For example, if the insured party makes sure its insurance company offers the best quotes, it would be an LP agent. Likewise, if AGFIL drops policy holders who engage in

fraud, AGFIL too would be LP.

Hierarchical VOs. Some VOs exhibit a hierarchical structure, naturally modeled in our architecture, based on delegation of commitments. For example, VO-2, which is formed among the entities AGFIL, Lee, the assessor, and the garage, has a hierarchical structure. In this case, AGFIL (R_1) has delegated the commitment of finding a garage, handling repairs, and preparing invoices to Lee (R_3). Here, R_3's PMP reports the events of contacting a garage, preparing estimates, and sending invoices to the PCCP of its parent entity R_1. R_3 has delegated the commitment of assessing the car damage repair estimate given by the garage (R_4) to the assessor (R_5). R_5 reports events of its assessment to R_3. Here, R_3 is committed to performing these goals for R_1. R_1 can perform continuous compliance checks based on these events sent by R_3. R_1 can impose policies on R_3 and override the decisions taken by R_3. For example, if R_1 realizes that R_3's policy of contacting an assessor only for estimates more than a limit (say, \$500), has allowed some fraud to go undetected, then it can override that policy by enforcing another policy that brings down the limit (to say, \$250) in R_3.

Definition 6 An agent A_i is *hierarchically proactive (HP)*, if it proactively enforces its policies on its child A_j by assimilating events sent by A_j. A_i's policies P_i may override A_j's policies P_j (written $P_i \triangleright P_j$).

Above, R_1 is an HP agent and $P_{R_1} \triangleright P_{R_3}$.

Definition 7 A delegation path q_e for a goal e, containing a HP-VO A_i and its child VO A_j, with $P_i \triangleright P_j$, and e not satisfying P_j, is complete, if the rest of the path without P_j is complete.

In the above example, the path: $\langle$VO-2 $\rightarrow R_1 \rightarrow R_3\rangle$ is complete when R_1 enforces a new policy, because R_1's policies override those of R_3.

Theorem 3 A VO goal can be accomplished and terminated even in the presence of HP agents, and their overriding policy enforcements if all the delegation paths are complete.

We know from Definition 7 that paths can be complete even in the presence of HP agents. When all paths (including paths having HP agents) are complete, the desired result directly follows from Theorem 1. In our example, the goals of VO-2 are accomplished even in the presence of an HP agent R_1.

5 Literature

We survey approaches based on the traditional policy architectures and some VO architectures that are relevant to our proposed approach.

Rei. The Rei [8] policy language includes constructs such as rights, prohibitions, obligations, dispensations, delegation, and revocation. The architecture underlying Rei presupposes a PDP-PEP representation. A resource manager functions like a PEP. It consults the Rei policy engine (functioning like a PDP), which produces a certificate specifying permissions, their validity period, and such. The proposed approach enhances such reactive policy mechanisms by offering a proactive policy mechanism that incorporates PMPs and PCCPs.

KAoS. KAoS is a collection of agent services that represent and reason about policies on a variety of platforms [13]. KAoS domain services enable agent-agent collaboration and external policy administration by capturing groups of agents, people, and other resources as organizations of domains and subdomains. KAoS policy services provide the specification and management of policies within these domains, including conflict resolution and enforcement. The domain manager ensures policy consistency and distributes them to components called *guards*, which interpret them à la PDPs. The enforcers take policies from the guards and enforce them in a platform-specific manner à la PEPs. The proposed approach is similar to KAoS in terms of using organizations. However, it provides a proactive policy architecture.

Utilization management, accountability, and security. Wasson et al.'s framework specifies and enforces VO policies in a centralized manner [14]. In their prototype, VOs consist of three main components: GateKeepers (representing the access points for resources), Enforcers (who carry out VO enforcement actions), and a Bank (collects resource utilization data). By contrast, our approach is decentralized enabling local sites or sub-VOs to specify and enact their policies, albeit in a manner that respects the constraints of higher-level VOs.

Communities. A community consists of members with similar objectives and similar resources to be shared. Feeney et al. support a nested community architecture with hierarchical policy enforcement, especially with respect to conflict resolution [4]. Proposed policies about a resource are checked for conflicts and recursively propagated to parent communities until they reach the community owning the resource, where they are deployed. Policy decisions at a community are made via consensus among its members. In our approach, a community can be modeled as a VO whose policies reflect the consensus of its members. The VO agent provides a locus for enforcing community policies and resolving conflicts among members.

Open distributed processing. RM-ODP is a reference model for open distributed processing [7]. RM-ODP models several independent concerns or *viewpoints*, of which

the *enterprise* viewpoint is relevant here. The enterprise viewpoint specifies a system and its environment as a *community* of (passive or active) objects formed for a purpose. For example, we may have a financial community consisting of people, banks, tellers, and such. An RM-ODP community resembles a VO in this respect, but it also admits purely virtual entities such as bank accounts, and even money. The enterprise viewpoint defines the VO members, their *roles*, and their policies. The proposed approach can be understood as an enhancement of RM-ODP as it provides a proactive policy-based architecture for managing the interactions between different VOs and their member agents.

6 Conclusion

This paper proposes a multiagent architecture for VOs that treats VOs as consisting of agents, potentially VOs in their own right. The nesting structure of VOs highlights the freedoms and constraints on the VOs at each level. The key advantages of this architecture are as follows.

Policy Modeling. The proposed architecture enables us to go beyond the traditional emphasis on reactive policies. A business not only must react to explicit requests (reactive) but also must monitor its environment, collate events, and determine how to act (proactive).

Policy Management. The proposed architecture recognizes that VOs are distributed. It supports two complementary perspectives. One is that there is a single locus of policy enforcement. The other is that a distributed organization must have parts that collaborate to enforce a given policy.

Relationships. The proposed architecture naturally supports complex nested structures. It supports managing the complementary properties of two VOs being unaware of each other's structure but gaining requisite visibility to interact effectively.

Social reasoning mechanisms play an important role in the design of agent architectures, enabling an agent to evaluate and reason about others using its dependencies with others [1]. Service relationships among different entities formed over a particular contract or a goal bring them together to form VOs. In an e-commerce setting, a service provider may depend on a consumer, or a consumer may depend on a provider. Dependencies are dynamic, because they can be formed and revoked at run time. Social relationships can form the basis of the policies of the entities in a VO. These service relationships among VOs and other enhancements to our formal VO definitions will be considered as future work. The relationships captured while describing VOs can dynamically change and becomes crucial for describing VO behaviors.

References

[1] R. Ashri, S. D. Ramchurn, J. Sabater, M. Luck, and N. R. Jennings. Trust evaluation through relationship analysis. In *Proc. of the 4th International Joint Conference on Autonomous Agents & Multiagent Systems*, pp. 1005–1011, 2005.

[2] CrossFlow/AGFIL. Insurance (motor damage claims) scenario. (Tech. Rep.), CrossFlow Consortium. 1999.

[3] C. L. Dumitrescu, M. Wilde, and I. Foster. A model for usage policy-based resource allocation in grids. In *Proc. 6th Intern. IEEE Wkshp Policies Distr. Syst. Netw. (POLICY)*, pp. 191–200, 2005.

[4] K. C. Feeney, D. Lewis, and V. P. Wade. Policy based management for Internet communities. In *Proc. 5th Intern. IEEE Wkshp Policies Distr. Syst. Netw. (POLICY)*, pp. 23–32, 2004.

[5] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of Supercomputing Applications*, 15(3), 2001.

[6] IETF WG. Terminology for policy-based management. *RFC 3198*, Nov. 2001.

[7] ISO/IEC 10746, ITU: X.901 - X.904. The Reference Model of Open Distributed Processing. 1995.

[8] L. Kagal, T. Finin, and A. Joshi. A policy language for a pervasive computing environment. In *Proc. 4th Intern. IEEE Wkshp Policies Distr. Syst. Netw. (POLICY)*, pp. 63–74, June 2003.

[9] OASIS. eXtensible access control markup language (XACML) version 2.0 specification document. *OASIS Standard*, Feb. 2005.

[10] M. P. Singh. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence & Law*, 7:97–113, 1999.

[11] M. P. Singh and M. N. Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. Wiley, 2005.

[12] Y. B. Udupi and M. P. Singh. Contract enactment in virtual organizations: A commitment-based approach. In *Proc. of the 21st National Conference on Artificial Intelligence (AAAI)*, 2006.

[13] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott. KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. In *Proc. 4th Intern. IEEE Wkshp Policies Distr. Syst. Netw. (POLICY)*, pp. 93–96, 2003.

[14] G. Wasson and M. Humphrey. Toward explicit policy management for virtual organizations. In *Proc. 4th Intern. IEEE Wkshp Policies Distr. Syst. Netw. (POLICY)*, pp. 173–182, 2003.

[15] R. Yavatkar, D. Pendarakis, and R. Guerin. A framework for policy-based admission control. *IETF WG – RFC 2753*, Jan. 2000.