Requirements for a Policy-Enforceable Agent Architecture

Travis D. Breaux and James M. Niehaus

Department of Computer Science
North Carolina State University
{tdbreaux, jmniehau}@eos.ncsu.edu

Abstract. Emerging legislation that governs consumer privacy presents a design challenge to multi-agent systems providing business, health-care and government services. As agents act on behalf of consumers and providers of goods and services, their compliance with laws governing information sharing and disclosure practices must be transparent and measurable to avoid prohibitive sanctions by regulators. Human-readable and machine-enforceable policies that govern agent behavior offer a promising avenue to safeguard against violations of law and achieve compliance in dynamic environments. We apply software engineering practices to this problem and present requirements for designing an agent-based policy language and agent framework and compare our approach to current practices. We elaborate our requirements using two scenarios demonstrating policy authorship and enforcement in a multi-agent environment. Our proposal is motivated by results from analyzing privacy policies and legislation related to services in e-commerce and health-care.

1 Introduction

Legislation such as the Health Insurance Portability and Accountability Act (HIPAA) and the Gramm-Leach-Bliley Act (GLBA) govern consumer privacy including the sharing of personal information necessary to do business in U.S. health-care and financial markets. Regulatory agencies such as the U.S. Department of Health and Human Services, Federal Trade Commission and Securities and Exchange Commission interpret both HIPAA and the GLBA to establish guidelines or rules with specific rights and obligations governing the behavior of businesses and consumers. These rights, obligations and rules have a direct impact on software requirements for systems operating in regulated markets.

Rights and obligations dictate the allowable and expected behavior of businesses and customers. Rights, authorizations or permissions all describe what parties *may* or *may not* do while obligations or commitments describe what parties *must* or *must not* do. For example, under the GLBA businesses may share consumer information with their affiliates. However, these businesses must also restrict sharing such information with non-affiliates. In addition, customers have the right to opt-out of certain information sharing practices. In general, rights and obligations mediate the impact of transactions as either protections or vulnerabilities to consumer's personal information.

Guidelines and rules both describe expected behavior for parties of an engagement. While these terms are often used interchangeably, guidelines are semantically broader than rules and often lack the details of how to handle specific situations. Rules on the other hand are generally more specific and therefore easier to understand. Rules clarify the contexts for assigning and invoking rights and enforcing obligations by adding conditions or restrictions to the circumstances of an engagement. Furthermore, rules govern a variety of business processes including specific software transactions within an organization, across organizations, and between an organization and consumers. Generally, businesses strive to develop policies that govern transactions between their employees, software systems and customers to assure compliance with both guidelines and rules established by regulators.

In addition to regulatory compliance, businesses create policies and coordinate their software systems in ways that maximize several goals including: efficiency, safety, reliability, security, internal auditing, traceability and ultimately a return on investment. The creation and maintenance of business software that follows organizational goals can be a time and resource intensive activity. Policies and actual software systems may be out of alignment due to limitations in the ability to configure and customize software. As a result, a business may have remarkably sound policies with poor actual business practices. Our goal is to reduce the misalignment between organizational policies and practice by bridging this gap with multi-agent systems.

The impact of non-compliance with regulations can be severe and largely intractable. For violations in the case of HIPAA, fines imposed by regulators range up to 250,000 USD and 10 years in prison. The extent of financial and legal liability established through lawsuits is simply intractable. However, Walker provides several social and economic arguments for understanding the costs of privacy [1] and Earp et al. have studied the meaning of privacy values to consumers [2]. Furthermore, Grzebiela identifies a relationship between the lack of both available insurance against loss of confidentiality and available mechanisms for ensuring accountability [3]. In other words, without available mechanisms to ensure accountability, businesses will not be able to insure against privacy losses. Understanding the risks and costs associated with developing non-compliant systems will increase motivation for integrating these mechanisms to become compliant.

System developers deploying agent-based systems whose transactions are governed by legislation such as HIPAA and the GLBA must demonstrate to regulators that their systems can sustain compliance transparency. Compliance transparency may be demonstrated first by system design and second through verifiable audit trails produced from actual transactions. The design should clearly demonstrate that control mechanisms are in place to guarantee that agents may exert their rights and fulfill their obligations. Alternatively, audit trails should confirm that governed transactions comply with regulatory rules. Demonstrating compliance will further establish and build trust between consumers, businesses, and government regulators in multi-agent systems.

We believe a multi-agent system design is well suited for policy enforcement and compliance transparency. In order to move this position forward, we are proposing several key challenges and introducing a set of requirements for a policy language and agent framework. We believe many organizations are facing these key challenges and that a policy-enforceable agent architecture will help address these challenges. In

section 2, we examine related work in software agents, policy-based systems and web services. In section 3, we present four key organizational challenges. In section 4 we discuss two scenarios for policy authorship and enforcement followed by section 5, where we present system requirements for a policy language and agent framework. In section 6, we discuss the benefits of this proposal with our conclusion in section 7 and references in section 8.

2 Background

Our approach to a policy-enforceable agent architecture relates to work distributed throughout several different domains including multi-agent systems (MAS), distributed systems management, and web services. Each of these areas contributes unique strengths to the literature and we highlight only contributions that demonstrate breadth and maturity. In each case, we identify strengths and briefly note shortcomings.

2.1 Multi-agent Systems

Multi-agent systems have made significant contributions in the theory of agency in software systems; specifically the theory regarding agent autonomy, collaboration and commitments. The popular Belief-Desire-Intention (BDI) model describes agent autonomy through notions of desire and open behavior. Policy-governed systems emphasize what agents *may do* (rights) and what agents *must do* (obligations). In contrast, the characterization of what agents *want to do* is always subject to rights and obligations. Furthermore, agent autonomy is increased through the assignment of rights and decreased through the assignment of obligations. In multi-agent systems, the *O-Autonomy* classification described by Carabelea et al. in which agent autonomy is restricted to norms including social laws and organizational structure [4] is similar to the type of autonomy expected in a policy-compliance framework.

Collaboration, an established agent systems goal, is more general than the notion of compliance, an established policy systems goal. Compliance restricts the freedom of activities shared by agents making collaboration more deterministic. In addition, desire and intentionality are irrelevant up to the moment of non-compliance; after which sanctions are imposed that include new obligations and/or the loss of rights. Ultimately, agents that do not comply are unable to participate in the system since they lack sufficient rights.

Commitments or obligations have been a critical issue in collaborative multi-agent systems. Castelfranchi examined their role in organizational structure and distinguished between social commitments that are shared between two agents and organizational commitments that are shared by agents in an organization [5]. Jain et al. take a different approach by introducing the sphere of commitment [6]. The sphere of commitment describes the scope of commitments for any one agent and provides for flexible integrity and data flow, autonomy for both providers and consumers, and flexible organizational structures. The approach by Jain et al. is insightful in the context of policy-governed transactions where commitments may simultaneously span several agents and organizations.

Rights and obligations have been formally defined in multi-agent systems. Fasli formalizes agent interactions including rights, obligations and sanctions [7]. While

Fasli's approach is based on the BDI model, the interactions between rights and obligations are relevant to policy governance. Fornara et al. propose semantics for an agent communication language to represent authorizations (rights) and commitments (obligations) inspired by speech acts [8]. Their proposal integrates communicative acts such as *propose*, *request*, *inform*, *promise*, *accept*, and *reject* with a definition of commitments to develop protocols for agent communication. However, there is also a need to represent temporality to relate events to each other; a capability that is missing from the proposed semantics. Dignum et al. provide a formal semantics for describing obligations and deadlines using temporal logic [9]. The FIPA Policies and Domains Abstract Architecture specification proposes scenarios and terms for an agent architecture [10]. Among these scenarios, the relationship between fulfilling obligations and reputation or accountability is presented; however, the lack of requirements and detail in the scenarios lessens this architecture's contribution to multi-agent systems, in general.

Agent negotiation is made possible through agent autonomy, so it is worth mentioning, here. Negotiation is a process of discovery to optimize benefits under accepted terms of authority. At a minimum, policies must be negotiated to remove conflicts; however, these types of negotiations may be performed by human users. In our approach, we demonstrate how conflicts can and should be resolved by human agents thus removing the complexity of this issue from the software agent paradigm in policy-governed transactions. In unregulated transactions not covered by our approach, negotiation may play a more important role if the parties to the transaction announce an interest in negotiating existing policies.

2.2 Distributed Systems Management

Distributed systems management includes policy languages to describe rights and obligations in network and security policies. Rights and obligations in this domain often associate users and systems with access control policies. In cases where the policy language is generalizable, the language includes the abstract notions of agency similar to multi-agent systems. In some approaches, policies are centralized in a repository as in Beigi et al. [11] while others use roles inspired by role-based access control or domains inspired by network management. Finally the Semantic Web has influenced policy languages with the Web Ontology Language (OWL) that combines theory from description logic with the Resource Description Framework (RDF) developed by the World-Wide Web Consortium (W3C) [12].

Damianou et al. introduce the Ponder language developed for network and security policy management [13]. Rights and obligations in Ponder were developed from an access control perspective; describing the subjects of rights and obligations using roles and domains. Montanari et al. build their Poema agent framework around the Ponder language [14]. Poema allows applications to be dynamically reconfigured using obligations specified in Ponder. Our approach advocates a more generalizable concept of agents than Poema or Ponder support. We believe roles and domains are helpful in classifying agents, however, they should not replace richer semantics for describing individual agents. The benefit of these semantics is evident when policies include references to business processes; a requirement in our approach. For a review of roles from the agent-perspective, see Odell et al. [15].

Tonti et al. advocate an ontological approach to policy languages in an OWL-based implementation called KAoS [16]. KAoS includes an ontology for representing agents in a policy-context including rights and obligations. Limitations recognized in KAoS include the difficulty for users to directly interface with OWL-encoded policies and the gap between expressions in OWL and application-level code. The general benefit is provided by OWL in terms of richer semantics for describing agents and their abilities beyond the expressivity of roles and domains in Ponder.

Kagal et al. introduce the policy ontology REI implemented in OWL for describing policies in pervasive systems [17]. Statements in REI are in part motivated by speech-acts making REI amenable to agent-based systems. REI provides delegation and conflict resolution using priorities, however, REI only allows delegating rights whereas obligations also require a form of delegation to third-parties. In addition, like KAoS, REI is description logic-based which is not accessible to non-technical stakeholders. Responding to this limitation will require the research and development of user interfaces that abstract away the language details of OWL.

2.3 Web Services

Web Services seek to promote the deployment of re-usable online services. The vision includes the ability to dynamically compose complex services from simple, distributed services. The challenge faced by service policies that describe and control dynamic service composition is shared by at least two different industry initiatives: the Web Services Policy Framework (WS-Policy) [18] and the Web Services Policy Language (WSPL) [19]. WS-Policy is extremely abstract with support for declaring general assertions and alternatives but no explicit semantics to describe agency or fine-grained rules. WSPL, on the other hand, is based on the eXtensible Access Control Markup Language (XACML) which provides a rich rule writing environment [20]. In XACML rules are built from targets, pre-conditions, and effects. The targets identify the elements governed by the rule and the pre-conditions specify constraints on those elements before the rule is activated. The effect is a Boolean value, either permit or deny. While XACML has the richest semantics for expressing rules, it lacks the notion of agency that coincides with rights, obligations and responsibilities. Furthermore, the effects are extremely limited whereas our policy analysis to-date shows that rule effects are as complex as the pre-conditions.

The Business Process Execution Language for Web Services (BPEL4WS) provides semantics for representing business processes and binding them to web service descriptions [21]. BPEL4WS can express simple activities and structured activities using constructs for sequences, switches and loops. BPEL4WS also interfaces directly to web service descriptions causing the high-level process descriptions to bind directly to application-level function descriptions. For multi-agent systems in general, Vidal et al. evaluate the effectiveness of BPEL4WS [22]. In our approach, the ability to bind business and application processes is necessary to demonstrate compliance.

3 Key Challenges

Organizations seeking transparent and verifiable compliance with policies face several challenges that must be addressed at the system requirements level. We propose

four challenges that are driven by the role of organizations, their analysts, other stakeholders, and emerging situations in policy-compliant systems.

Challenge 1. Maturing organizations must transfer their informal, rote human decisions into formal, repeatable software decisions. Efficiency and predictability are two desirable goals in policy-compliant systems. In an organization, these goals are accomplished through automation. In software engineering, automating business processes requires identifying human-driven tasks that are predictable and well-understood and formalizing these tasks into repeatable software processes. Furthermore, software-driven processes are easier to instrument than human-driven tasks for compliance auditing and testing.

Challenge 2. In order to formalize and audit organizational behavior, tacit knowledge must be made explicit. Tacit knowledge refers to the unwritten experience people acquire from new and recurring encounters. In an organization, analysts employ tacit knowledge when interpreting and applying policies to emerging situations. These interpretations may challenge or extend existing policies without ever being formalized or providing feedback to such policies. In order to effectively audit an organization, tacit knowledge must be captured and coordinated with policy at the moment people are encountering these new situations.

Challenge 3. Many situations are either too complex or unpredictable for automated reasoning; requiring the integration of human-in-the-loop style decision making processes. Effective policies must be predictive of future events. As organizations expand operations or shift focus, policies may be challenged by unforeseen encounters. For this reason, compliance is an evolutionary process that requires continual intervention by analysts to assure that policies are consistent across new and emerging situations.

Challenge 4. Provide minimal policy specification required to deploy software systems. An extreme alternative to minimally specified policies is a complete specification that attempts to characterize all possible encounters. While it is desirable to have the most robust policies possible, this extreme is idealistic and in general holds policies to a higher standard than the criteria for testing and developing software. Furthermore, policy elaboration and evolution is an online process affected by real encounters. Complete policy specification is overly cumbersome and often impossible given this situation. With humans-in-the-loop, minimal specification to deploy encourages compliance with critical policies while providing sufficient flexibility for organizations to adapt and grow their policy infrastructure with emerging conditions.

4 Policy Use Scenarios

Two scenarios demonstrating policy authorship and enforcement are presented that elaborate interactions between human and software agents in a policy-enforceable agent architecture while avoiding specific implementation details. Each scenario provides insight into the requirements for a policy language and agent framework later specified in section 5. In the following two scenarios, policies are collections of rules and each policy contains at least one rule. Therefore, the term policy may be used interchangeably with rule.

4.1 Policy Authorship

In the policy authorship scenario illustrated in Figure 1, policies are authored and introduced into the policy-base. In order to ensure consistency and accountability, each new policy must be investigated to associate source rules with derived rules and identify and resolve conflicting policies before the policy can be distributed to agents.

Rules in a new policy may be derived from high-level rules (guidelines) or mid-level rules in existing policies. In this case, the existing or source rule partially justifies the new rule and we must explicitly account for this association. Challenges to rules that occur during policy enforcement may rely on these associations in order to justify whether or not to grant exceptions. For example, challenging a source rule indirectly challenges all of the rules that were derived from the source rule. In addition, challenging a derived rule indirectly challenges the source rule.

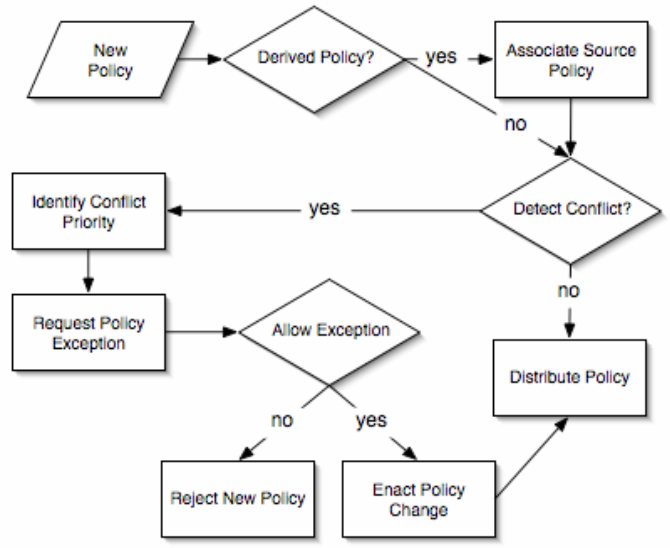

Figure 1: Policy Authorship Scenario

After associating a rule with any source rules, conflicts within a policy or between policies must be identified. Conflicts occur when the pre-conditions in two policy rules intersect yet their effects conflict. When no conflicts are detected, a policy is ready for distribution. Otherwise, a conflict resolution mechanism must be applied.

Resolving a conflict between a new policy rule and an existing rule is easily done if one of the two rules has priority over the other. In this case, an exception is made to the lower-priority policy. The intersection of the pre-conditions (e.g., the conflict) will be removed from the pre-condition of the lower-priority rule. For example, consider two rules R_1 and R_2 and their associated pre-conditions p_1, p_2 and effects e_1, e_2. In addition, let R_1 have a higher priority that R_2. If $p_1 \cap p_2 \neq \emptyset$ and e_1 conflicts with e_2, then the lower priority rule R_2 will have new pre-conditions $(p_1 - p_2)$. If the $p_1 \cap p_2 = p_2$ then we say rule R_1 completely overrides rule R_2 in which case $\underline{R_2}$ will be ineffec-

tive and should be removed from the system. In this way, the higher-priority rule is unaffected and the source of the conflict is removed. If priorities are not maintained between policies themselves, they must be established through online mediation between the human agents who are responsible for maintaining each conflicting policy.

It is important to clarify that policy authorities are the people responsible for maintaining a policy during its lifetime. Policy authorities and policy authors may be different people and policy authorities may not have equal rights to change policies, either. For example, an authority may be allowed to make temporary exceptions to a specific policy on a case-by-case basis but they may not be permitted to change the policy permanently.

4.2 Policy Enforcement

In the policy enforcement scenario illustrated in Figure 2, everyday events occur that must be handled by a software agent using available policy rules. In a relatively stable environment, most events will be predictable and policies will exist to handle them. These policies are called hard policies, since they are consistently applied and are not susceptible to new exceptions. However, if an event does not match the precondition of a policy rule it must be escalated to a human agent for evaluation. There are other reasons an event may be escalated for evaluation by a human agent including the need for remediation that is preferably handled by human agents.

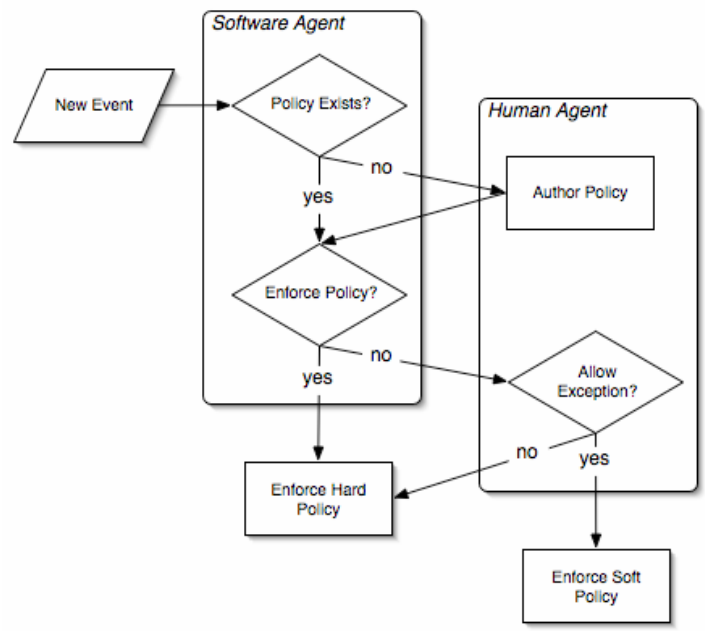

Figure 2: Policy Enforcement Scenario

Thus the human agent may receive an escalated event for two reasons: 1) if no policy rules apply to the specific event or 2) the event was escalated in spite of an exist-

ing policy. In the first case, the human agent should author a policy to handle the event or refer the event to an authority capable of authoring such a policy. The newly authored policy will enter the system and the event will be re-evaluated in the context of the new policy. In the second case where a policy exists, the human agent must evaluate the context of the event and determine if an exception is warranted. If an exception is undesirable, then the existing policy will be applied to the event. Otherwise, the human agent must author an exception to the existing policy to handle the event.

Exceptions to an existing policy may be temporary such as short-duration or one-time exceptions. For example, a policy may limit resource usage between certain business hours to a specific purpose while an employee may need to break this policy to complete a high priority report for another purpose. Such exceptions are instantiated by developing a policy rule with pre-condition that identifies a subset of events matching the existing rule and limited to only the events for which the exception is relevant. The exception is handled in the same manner for resolving conflicts described in section 4.1. The event subset includes the new event that triggered the exception but may generalize to other events as well if the event is related to a class of exceptions. The new rule, however, will have different effects than the existing rule and hence the cause for requiring the exception. The rules that are more susceptible to temporary exceptions are called soft policies.

5 Requirements

We propose requirements for a policy-enforceable agent architecture that are derived from scenarios in section 4 and the analysis of privacy policies [23]. In our proposed requirements, we distinguish human and software agents. Human agents are decision makers from across an organizational hierarchy: from lower-tier business analysts to upper-tier chief privacy and information officers. In terms of policies, software developers are stakeholders in the development and maintenance of the systems but are not referenced by the policy per se. Software agents include application-level software processes with abilities (functions) that send and receive specific data objects. Human and software agents both have abilities called actions. Tasks and activities are descriptions of agents performing actions. Events are tasks or activities performed in a time-relevant manner. Tasks and activities alone are considered timeless, however. The requirements are separated into two sections: the policy language in section 5.1 and the agent framework in section 5.2. Throughout the following requirements, we provide relevant examples from legislation such as HIPAA and the GLBA regarding information practices.

5.1 Policy Language

In a policy-enforceable agent architecture, the policy language provides the necessary abstractions to coordinate policies with system functionality. The policy language serves as an interface between various stakeholders and the runtime software components in multi-agent systems. As an interface, the policy language is a true agent communication language used by both human and software agents. The requirements numbered *PR1* through *PR8* are functional requirements while the requirement *PN1* is a non-functional requirement. *PR1* through *PR5* are requirements for expressing

agent, object, and event-level abstractions. *PR6* through *PR8* are requirements that address policy-level abstractions.

***PR1**: The policy language shall express activities using agents and their abilities to affect objects and other agents through actions.*

Policies govern the behavior of agents and include references to both agent capabilities and the states of objects and agents. However, agents are distinguished from objects at least by their capability to perform actions. For example, the GLBA requires financial institutions to protect information collected from individuals. In this example, the agent collectives "financial institutions" must have the capabilities "to protect" and "to collect" the object "information." In order to enforce specific protections, the policy language must explicitly define what information objects are collected and protected and which agents are providing and collecting this information.

***PR2**: The policy language shall express activities for both human and software agents including business processes.*

The capability of software to implement business processes will change with emerging technology and evolving business practices. Therefore, equally supporting the expression of both human and software agent activities allows system designers the flexibility to decide which business processes should be implemented by humans versus software. Furthermore, should new technological capabilities come into existence, designers may develop a migration plan to transfer business processes from human agents to software agents. These extensions must require only subtle changes to policies encoded in the policy language.

***PR3**: The policy language shall express deontic relationships identifying agent abilities as rights or obligations.*

Effective policies attribute activities to agents as rights or obligations. Breaux and Antón have shown that rights and obligations can be expressed in terms of actions performed by agents [23]. Rights or permissions *allow* agents to perform actions while obligations *require* agents to perform actions. In GLBA for example, customers have the right to opt-out of certain information sharing practices. In addition, financial institutions are obligated not to share their customer's financial information with non-affiliates. Rights and obligations may be conditionally or unconditionally asserted depending on contextual information. For example, financial institutions may not be required to protect information collected from voluntary surveys if the survey is not part of a business transaction.

***PR4**: The policy language shall express both guidelines and rules that govern business processes.*

It is important that the policy language describe guidelines and rules with a pre-condition and desired effect. If the effect is unconditional, then the pre-condition may be omitted. Pre-conditions and effects both include references to events and the states of agents and objects. In HIPAA for example, the health-care provider must provide privacy notices to patients upon request. Health-care providers include clinics as well as emerging e-services such as online pharmacies and health insurance portals. In rule parlance, the pre-condition would include the agent "patient" performing the action

"requests the privacy notice." The effect is the agent "health-care provider" performing the action "provide the privacy notice to the patient."

***PR5**: The policy language shall express temporal and periodic relationships between activities and object states.*

Events describe activities and states of agents and objects with respect to absolute and relative time. Absolute time is a fully-qualified calendar time, such as "Today at 9:00 am" or "July 25th, 2005." Relative time refers to the time between events, such as "sometime before" – or "10 hours after" – an event. Time relative to an absolute time is absolute such as "10 minutes before 5:00 pm" is simply to "4:50 pm." Event descriptions using temporal relations often occur in rules. In HIPAA for example, requests for access to personal health information by patients must be honored within 30 days of the request. In the GLBA, new customers are permitted to opt-out of information sharing practices within 30 days of their first transaction.

***PR6**: The policy language shall express application-level functions as either events or objects.*

Application-level functions are used to affect change within the system and receive feedback on the current system state. Affecting change through rule effects requires functions to initiate change within the system. Aligning rule pre-conditions with system states and events requires functions to acquire these states and events before or during the evaluation of pre-conditions. In both cases, functions must be bound to language-level expressions. For example, if the policy dictates the agent "health-care provider" will perform the action "communicate with the patient using SSL" then the system must respond by encrypting the communication using the Secure-Socket Layer. Furthermore, an object representing the current time should be bound to the system clock or a valid time service.

***PR7**: The policy language shall express agent responsibilities for policy evaluation and enforcement.*

Policy evaluation includes the human abilities to author, maintain and update policies. Policy enforcement includes both human and software agent abilities to enforce the effects of a policy rule given a satisfied pre-condition. The language should allow authors to clearly identify which agents are responsible for enforcement and evaluation. Section 4.2 describes a scenario where both of these responsibilities are employed to complete a transaction.

***PR8**: The policy language shall maintain traceability between high-level policies that justify low-level policies.*

For example, there is a need to express traceability from the high-level policy "all systems must be secure" to the mid-level policy "all systems must encrypt communication" to the low-level policy "all systems must use SSL for socket connections." The higher-level policies serve as the justification for deriving lower-level policies. The justification or purpose of a policy is important when human agents must make policy evaluation decisions such as whether or not to allow exceptions to existing policies or whether to ignore new policies that conflict with existing policies.

***PN1**: The policy language must be human-readable and machine-enforceable.*

The policy language will be used by a variety of stakeholders (technical and non-technical) to author and evaluate policies; therefore the policy language semantics must be accessible to both human and software agents. The language should consider conventional means by which human agents communicate policy rules (e.g., natural language, speech acts) in addition to standards for developing software such as Application Programmer Interfaces (APIs) or remote-procedure calls (RPCs).

5.2 Agent Framework

The agent framework is responsible for coordinating interactions between agents, events, and policies. The framework is intended to be light-weight by providing only minimal support for a policy-enforceable agent architecture, while not restricting the agents' reasoning style or abilities. In particular, these functional requirements refer to mechanisms that handle policy delegation, exceptions, conflicts, subscriptions, and general accountability.

***AR1**: The agent framework shall support conditional delegation of rights and obligations.*

Agents may be permitted to delegate rights and/or obligations to other agents conditionally due to certain events or system states. These delegations may persist indefinitely or only for the duration of the event or system state. In GLBA for example, financial services may extend access rights to their affiliates baring any opt-out requests by customers. In the event of an opt-out request, the delegation of these rights must be revoked on a per-request basis. In HIPAA on the other hand, health-care providers are obligated to limit access and disclosure of protected information. These providers must also ensure that their affiliates with whom they share protected information accept these same obligations. In this case, providers must have the ability to delegate obligations to affiliates.

***AR2**: The agent framework shall support temporary policies such as short-term and one-time exception policies.*

In general, a policy agent framework must be sufficiently robust to handle a variety of events but flexible enough to support unpredictable situations. Therefore, policies may be adopted temporarily based on the duration of specific events, system states, or time periods. Furthermore, exceptions to policies may be granted on a case-by-case basis. The framework must support enforcement of temporary policies and the removal of these policies upon expiration. Section 4.2 illustrates a scenario where a temporary exception may be required to complete a transaction.

***AR3**: The agent framework shall support policy conflict identification and resolution.*

Conflicts occur when the pre-conditions for two policy rules match but have incompatible effects. Resolving such conflicts may require ignoring one of the two conflicting rules, re-conditioning the rules, etc. The resolution mechanism may be automatic if rules have established priorities or may involve decisions from human agents

responsible for the conflicting rules. Section 4.1 illustrates a scenario when a conflict arises and a priority-based mechanism is applied to achieve an acceptable resolution.

***AR4**: The agent framework shall support policy subscription for restricting the distribution of policies and policy change notifications to authorized parties, only.*

It is a security requirement to limit the distribution of policies only to concerned parties. For example, agents that enforce or evaluate policies are always policy subscribers. Agents whose rights or obligations interact with policy rules are subscribers to those policies as well. Agents that generate legitimate events will indirectly interact with policies affected by these events; however, they may not be subscribers to those policies.

***AR5:** The agent framework shall support traceability across policy authorship, enforcement and evaluation.*

The agent framework must record enforcement decisions by software agents and authorship events and evaluation decisions by human agents. Policy evaluation includes both the decisions to enforce hard policies and soft policies; the latter by creating policy exceptions. Included in the traceability for each of the decisions is the ability to exercise rights and fulfill obligations. Accounting for these decisions is necessary to ensure the agent architecture maintains compliance transparency for an organization.

6 Discussion and Benefits

Policy-enforceable agent systems that satisfy our requirements offer unique views in multi-agent systems and several benefits to the sustainability of organizations. In multi-agent systems, we highlight the benefits to understanding autonomy and trust. For organizations, policy-enforceable systems provide stability, accountability, predictability, security and improved human-computer interactivity.

6.1 Benefits to Multi-agent Systems

Policy-enforceable agent systems include a unique view on agent autonomy. Policy-enforceable agent systems do not remove agent autonomy nor do they reduce the agents to a collection of objects. Alternatively, autonomy exists within the confines of rights, obligations, and responsibilities imposed by participating in the system. While rights and obligations limit an agent's abilities, an agent still may consider exercising different rights to solve a problem in different ways – optimizing individual benefits under the constraints of rights and obligations.

Trust in policy-enforceable agent systems concerns the trust people place in these systems. If people can access and understand the policies that govern agent systems, they can begin to build expectations for transactions and evaluate system performance based on their individual expectations. Accounting for an agent system's ability to transparently comply with policies will further instill a sense of reputability among agents with users and multi-agent systems.

6.2 Benefits to Organizations

As the size of the policy-base increases, the interactive requirement for human agents decreases. For each new encounter, a human agent will contribute a new decision to the policy-base. Each decision will represent a class of encounters and both formally identify a situation (via rule pre-conditions) and determine the desirable outcomes (via rule effects). For subsequent encounters, the decision will face challenges that will harden it against – or soften it to support – exceptional circumstances. As the number of encounters increases, the policy-base will grow and better characterize the most common transactions. As a result, human agents will be reserved for handling only new and exceptional circumstances that occur less frequently over time.

Policy-decisions are traceable and accountable. For each encounter, a set of formal rules are invoked based on specific details relevant to the situation. The human and software agents responding to the encounter, the sequence of invocations, and the details relevant to each invocation may persist for the purpose of auditing transactions. Furthermore, the responsible agents for authoring and maintaining each rule are quickly and easily identifiable. This level or traceability is important to understand both *why* and *how* decisions are made in complex, dynamic multi-agent environments.

Policy structure is hardened for predictable, stable contexts while softened (made flexible) for unpredictable, dynamic contexts. As decisions are exercised in formal rules, challenges will be presented by other policy authors or by new and unforeseen encounters. Decisions that are initially too broad will accept challenges and make exceptions to unforeseen situations. The process of introducing exceptions will soften the original decision, making it more robust and therefore survivable. On the other hand, decisions that withstand challenges with few exceptions will harden. Different stakeholders have different priorities in terms of hard versus soft policies. Regulators will be reassured if the legislated agenda is implemented by hard policies. Businesses and consumers will benefit from exceptions that enable the completion of more transactions. Our approach allows both sets of stakeholders to understand and analyze the trade-offs between hardening and softening the decision space through policies.

Access to policies is restricted to relevant parties, only. Policies encode both public knowledge, such as governmental regulations, and private knowledge, such as tacit knowledge and internal business practices. Since each rule includes the detailed circumstances under which it is invoked, it is easy to associate only contextually relevant rules with users. Furthermore, agents will only have access to policies that they are responsible for enforcing and maintaining. Restricting access to policies will limit the scope of visible policies to only those required for effective decision making and further secure organizational knowledge against unauthorized access.

The agent framework supports multi-tier organizational structure and human-in-the-loop decision making. The framework assigns responsibility for enforcing policies to software agents while keeping responsibility for authoring and maintaining policies with human agents. Furthermore, the framework and policy language both maintain relationships of authority between policies, authors, and authorities (maintainers). These hierarchical relationships support and are motivated by the organizational structure, better aligning software processes with business processes.

7 Conclusion

Multi-agent systems that operate in regulated environments face important compliance challenges. An approach to addressing these challenges involves developing a policy-enforceable agent architecture. We propose two scenarios to elaborate issues in policy authorship and enforcement from which we derive several system requirements for a policy-enforceable agent architecture. The proposal uses a unified model of business and software processes in which human and software agents collaborate to address key challenges and provide much-needed benefits to an organization. The proposal is motivated by analysis of privacy policies and law that govern attractive application areas for multi-agent systems including e-commerce and health care.

8 References

1. Walker, K. "The Costs of Privacy" Harvard Journal of Law and Public Policy, v. 25, no. 1, 2001, pp. 87 – 129.
2. Earp, J. B., Poindexter, J. C., Baumer, D. L. "Modeling Privacy Values with Experimental Economics" *In Proc. of the ACM Workshop on Privacy in Electronic Society (WPES)*, Washington, D.C., USA, Oct. 2004, pp. 25 – 27.
3. Grzebiela, T. "Insurability of Electronic Commerce Risks" *In Proc. of the 35th Hawaii Int'l Conf. on Sys. Sci. (HICSS-35)*, Waikoloa, Hawaii, USA, Jan. 2003, pp. 185
4. Carabelea, C., Boissier, O., Florea, A. "Autonomy in Multi-agent Systems: A Classification Attempt" In Proc. *1st Int'l Workshop on Computational Autonomy*, Melbourne, Australia, Jul. 2003. LNCS, v. 2969, pp. 103 – 113.
5. Castelfranchi, C. "Commitments: From Individual Intentions to Groups and Organizations" *In Proc. 1st Int'l Conf. on Multi-Agent Systems*, San Francisco, CA, USA, Jun. 1995, pp. 44 – 48.
6. Jain, A. K., Aparico IV, M., Singh, M. P. "Agents for Process Coherence in Virtual Organizations" In Comm. of the ACM, v. 42, n. 3, Mar. 1999, pp. 62 – 69.
7. Fasli, M. "Social Interactions in Multi-agent Systems: a Formal Approach" In Proc. Int'l Conf. on Intelligent Agent Tech. (IAT'03), Oct. 2003, pp. 240 – 246.
8. Fornara, N., Colombetti, M. "Defining Interaction Protocols Using a Commitment-based Agent Communication Language." *In Proc. of the 2nd Joint Int'l Conf. on Auto. Agents and Multi-agent Systems*, Melbourne, Australia, pp. 520 – 527.
9. Dignum, F., Broerson, J., Dignum, V., Meyer, J-J. "Meeting the Deadline: Why, When and How" *In Proc. of the 3rd Workshop on Formal Approaches to Agent-based Sys. (FAABS'04)*, Greenbelt, MD, Apr. 2004, LNCS, v. 3228, pp. 30 – 40.
10. FIPA Policies and Domains Abstract Architecture Specification (PC00089B). Published by Foundation for Intelligent Physical Agents (FIPA). Jan. 2003.
11. Beigi, M., Calo, S., Verma, D. "Policy Transformation Techniques in Policy-based Systems Management" *In Proc. of the IEEE 5th Int'l Workshop on Policies for Distributed Systems and Networks (POLICY'04)*, Yorktown Heights, NY, USA, Jun. 2004, p. 13 – 22.

12. Bechofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuiness, D., Patel-Schneider, P. F., Stein, L. A. "OWL Web Ontology Language Reference", http://www.w3.org/TR/owl-ref/
13. Damianou, N., Dulay, N., Lupu, E., Sloman, M., "The Ponder Specification Language," *In Proc. of the Int'l Workshop on Policies for Distributed Systems and Networks (POLICY'01)*, Bristol, UK, Jan. 2001, pp. 29-31.
14. Montanari, R., Lupu, E., Stefanelli, C. Policy-based Dynamic Reconfiguration of Mobile-Code Applications." IEEE Computer, v. 37, no. 7, Jul. 2004, pp. 73 – 80.
15. Odell, J., van Dyke Paranuk, H., Fleischer, M. "The Role of Roles in Designing Effective Agent Organizations" In Proc. *2nd Int'l Workshop on Soft. Engr. for Large-scale Multi-agent Sys. (SELMAS'03)*, Portland, OR, May 2003, pp. 27 – 38.
16. Tonti, G., Bradshaw, J., Jeffers, R., Montanari, R., Suri, N., Uszok, A. "Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder" *In Proc. of the 2nd Int'l Semantic Web Conference (ISWC'03)*. LNCS, Springer-Verlag, Sep. 2003, pp. 419 – 437.
17. Kagal, L., Finn, T., Joshi, A. "A Policy Language for a Pervasive Computing Environment" *In Proc. of the 4th Int'l Workshop on Policies for Distributed Systems and Networks (POLICY'03)*, Jun. 2003, pp. 63 – 74.
18. Bajaj, S., Box, D., Chappel, D., et a. "Web Services Policy Framework (WS-Policy)" Published online by BEA, IBM, Microsoft, et al., Sep. 2004, http://www-106.ibm.com/developerworks/library/ws-polfram/
19. Anderson, A. H. "An Introduction to the Web Services Policy Language (WSPL)" *In Proc. Int'l Workshop on Policies for Distributed Systems and Networks (POLICY-04)*, Yorktown Heights, NY, Jun. 2004, pp. 189 – 192.
20. Moses, T., ed. "eXtensible Access Control Markup Language (XACML) Version 2.0," Dec. 2004, Published by OASIS, http://docs.oasis-open.org/xacml/access_control-xacml-2_0-core-spec-cd-04.pdf
21. Andrews, T., Curbera, F., Dholakia, H., et al. "Business Process Execution Language for Web Services (BPEL4WS)", Published by IBM, May 2003, ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf
22. Vidal, J. M., Buhler, P., and Stahl, C. "Multiagent Systems with Workflows." *IEEE Internet Computing*, v. 8, no. 1, pp. 76 – 82, Jan. 2004.
23. Breaux, T. D., Antón, A. I. "Deriving Semantic Models from Privacy Policies," In Proc. *IEEE 6th Int'l Workshop on Distributed Systems and Networks (POLICY-2005)*, Stockholm, Sweden, Jun. 2005.