Chapter 12
Making Agents Acceptable To People

Jeffrey M. Bradshaw(1), Patrick Beautement(2), Maggie R. Breedy(1), Larry Bunch(1), Sergey V. Drakunov(3), Paul J. Feltovich(1), Robert R. Hoffman(1), Renia Jeffers(1), Matthew Johnson(1), Shriniwas Kulkarnt(1), James Lott(1), Anil K. Raj(1), Niranjan Suri(1), Andrzej Uszok(1)

(1) *Institute for Human and Machine Cognition, University of West Florida, USA*
(2) *QinetiQ, Malvern Technology Centre, UK*
(3) *Department of Electrical Engineering and Computer Science, Tulane University, USA*

Abstract. Because ever more powerful intelligent agents will interact with people in increasingly sophisticated and important ways, greater attention must be given to the technical and social aspects of how to make agents acceptable to people [87]. The technical challenge is to devise a computational structure that guarantees that from the technical standpoint all is under control. The social challenge is to ensure that agents and people interact gracefully and to provide reassurance to people that all is working according to plan. In this chapter, we outline our efforts to address some of these technical and social concerns through the use of a policy-based approach as implemented in the KAoS framework. From a technical perspective, we want to be able to help ensure the protection of agent state, the viability of agent communities, and the reliability of the resources on which they depend. To accomplish this, we must guarantee, insofar as is possible, that the autonomy of agents can always be bounded by explicit enforceable policy that can be continually adjusted to maximize the agents' effectiveness and safety in both human and computational environments. From a social perspective, we want agents to be designed to fit well with how people actually work together. Explicit policies governing human-agent interaction, based on careful observation of work practice and an understanding of current social science research, can help assure that effective and natural coordination, appropriate levels and modalities of feedback, and adequate predictability and responsiveness to human control are maintained. These factors are key to providing the reassurance and trust that are the prerequisites to the widespread acceptance of agent technology for non-trivial applications.

1 Introduction

Since the beginning of recorded history, people have been fascinated with the idea of non-human agencies.[1] Popular notions about androids, humanoids, robots, cyborgs, and science

[1]Works by authors such as Schelde [95] and Clute and Nicholls [33] who have chronicled the development of popular notions about androids, humanoids, robots, and science fiction creatures, are a useful starting point for software agent designers wanting to plumb the cultural context of their creations. Lubar's chapter "Information beyond computers" in [77] provides a useful grand tour of the subject. See Ford, Glymour, and Hayes [50] for a delightful collection of essays on android epistemology.

fiction creatures permeate our culture, forming the backdrop against which software agents are perceived. The word robot, derived from the Czech word for drudgery, became popular following Karel Capek's 1921 play *RUR: Rossum Universal Robots* [26] (Figure 1). While Capek's robots were factory workers, the public has also at times embraced the romantic dream of robots as "digital butlers" who, like the mechanical maid in the animated feature *The Jetsons* would someday putter about the living room performing mundane household tasks (Figure 2).[2] Despite such innocuous beginnings, the dominant public image of artificially intelligent embodied creatures has often been more a nightmare than a dream. Would the awesome power of robots reverse the master-slave relationship with humans (Figure 3)?[3] Would seeing the world through the eyes of agents lead to dangerously distortions of reality (Figure 4)? Everyday experiences of computer users with the mysteries of ordinary software, riddled with annoying bugs, incomprehensible features, and dangerous viruses reinforce the fear that the software powering autonomous creatures would pose even more problems. The more intelligent the robot, the more capable of pursuing its own self-interest rather than that of its human masters (Figure 5); the more human-like the robot, the more likely it is to exhibit human frailties and eccentricities (Figure 6). Such latent images cannot be ignored in the design of software agents-indeed, there is more than a grain of truth in each of them!

"Agents occupy a strange place in the realm of technology," summarizes Don Norman, "leading to much fear, fiction, and extravagant claims" [87, p. 51]. By their ability to operate independently without constant human supervision, they can perform tasks that would be impractical or impossible using traditional software applications. On the other hand, this additional autonomy, if unchecked, also has the potential of effecting severe damage if agents are poorly designed, buggy, or malicious. Because ever more powerful intelligent agents will increasingly differ software that people are accustomed to, we need to take into account social issues no less than the technical ones if the agents we design and build are to be acceptable to people:

"The technical aspect is to devise a computational structure that guarantees that from the technical standpoint, all is under control. This is not an easy task.

---

[2]It is interesting to note that today's robotic vacuum cleaners have little resemblance to mechanical maids. However that is true in part because they are conceived as inexpensive single-function appliances and not multi-purpose assistants. Were our current technical prowess sufficient to build cheap, smart, and versatile robotic assistants, there is little doubt that we would prefer models that featured a "good brain and an unspecialized body" [83, p. 489]. See also Section 4.4 below.

[3]Whether or not such futures are plausible is beside the point—there is no doubt that the fears are real for many people right now. For example, Bill Joy notes the "prophecy" of the Unabomber, asserting that while his "mentality was criminal, his vision is rather realistic": 'What we do suggest is that the human race might easily permit itself to drift into a position of such dependence on the machines that it would have no practical choice but to accept all of the machines' decisions. As society and the problems that face it become more and more complex and machines become more and more intelligent, people will let machines make more of their decisions for them, simply because machine-made decisions will bring better results than man-made ones. Eventually a stage may be reached at which the decisions necessary to keep the system running will be so complex that human beings will be incapable of making them intelligently. At that stage the machines will be in effective control. People won't be able to just turn the machines off, because they will be so dependent on them that turning them off would amount to suicide.' *Theodore Kaczynski* - the criminal Unabomber. On the other hand, just one year ago Stephen Hawking, the noted physicist, suggested using genetic engineering and biomechanical interfaces to computers in order to make possible a direct connection between brain and computers 'so that artificial brains contribute to human intelligence rather than opposing it.' The professor concedes it would be a long process, but important to ensure biological systems remain superior to electronic ones. 'In contrast with our intellect, computers double their performance every 18 months,' he told Focus magazine. 'So the danger is real that they could develop intelligence and take over the world."[66].

Figure 1: Scene from Capek's play *Rossum Universal Robots* [25, p. 19].

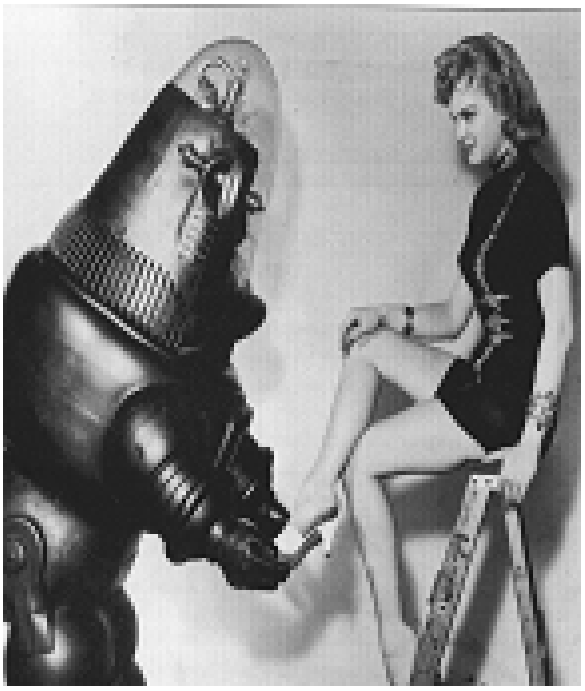

Figure 2: Electro the Robot (aka Robby the Robot) as digital butler to Anne Frances [77, p. 376].

Figure 3: Powerless in the grasp of a robot (From *Astounding Science Fiction*, October 1953 [77, p. 383]).

Figure 4: Select-O-Vision [113].

Figure 5: A robot thief [25].

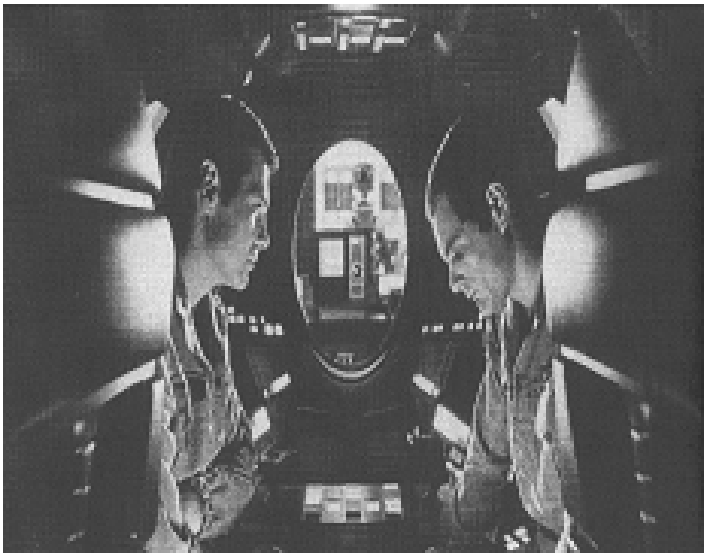

Figure 6: In Arthur C. Clarke's *2001: A Space Odyssey*, humans are compelled to hide from the psychotic computer HAL [95, p. 144].

The social part of acceptability is to provide reassurance that all is working according to plan.... This is also a non-trivial task" [87, p. 51].[4]

This chapter summarizes our efforts to address, through a policy-based approach (Section 2), some of the technical and social aspects of agent design for increased human acceptability. From a technical perspective, we want to be able to help ensure the protection of agent state, the viability of agent communities, and the reliability of the resources on which they depend. To accomplish this, we must guarantee, insofar as is possible, that the autonomy of agents can always be bounded by explicit enforceable policy that can be continually adjusted to maximize the agents' effectiveness and safety in both human and computational environments (Section 3).

From a social perspective, we want agents to be designed to fit well with how people actually work together. Explicit policies governing human-agent interaction, based on careful observation of work practice and an understanding of current social science research, can help assure that effective and natural coordination, appropriate levels and modalities of feedback, and adequate predictability and responsiveness to human control are maintained (Section 4). In short, interaction among humans and agents must be graceful and should enhance rather than hinder human work. All these factors are key to providing the reassurance and trust that are the prerequisites to the widespread acceptance of agent technology for non-trivial applications. [5]

2 Addressing Agent Acceptability Through the Use of Policy

The idea of building strong social laws into intelligent systems can be traced at least as far back as the 1940s to the science fiction writings of Isaac Asimov [6]. In his well-known stories of the succeeding decades he formulated a set of basic laws that were built deeply into the positronic-brain circuitry of each robot so that it was physically prevented from transgression. Though the laws were simple and few, the stories attempted to demonstrate just how difficult they were to apply in various real-world situations. In most situations, although the robots usually behaved "logically," they often failed to do the "right" thing—typically because the particular context of application required subtle adjustments of judgments on the part of the robot (e.g., determining which law took priority in a given situation, or what constituted helpful or harmful behavior).[6]

Shoham and Tennenholtz [99] introduced the theme of social laws into the agent research

[4]Similarly, Alan Kay has written: "It will not be an agent's manipulative skills, or even its learning abilities, that will get it accepted, but instead its safety and ability to explain itself in critical situations. At the most basic level the thing we want most to know about an agent is not how powerful it can be, but how trustable it is" [68, pp. 205-206].

[5]A more complete study of many of these topics can be found in [15] For an entertaining and informative general characterization of various approaches to human-centered computing, see [61].

[6]In an insightful essay, Roger Clarke explores some of the implications of Asimov's stories about the laws of robotics for information technologists [32]. Weld and Etzioni [120] were the first to discuss the implications of Asimov's first law of robotics for agent researchers. Like most norm-based approaches described below (and unlike most policy-based approaches) the safety conditions are taken into account as part of the agents' own learning and planning processes rather than as part of the infrastructure. In an important response to Weld and Etzioni's "call to arms," Pynadath and Tembe [91] develop a hybrid approach that marries the agents' probabilistic reasoning about adjustable autonomy with hard safety constraints to generate "policies" governing the actions of agents. The approach assumes a set of homogeneous agents who are motivated to cooperate and follow optimally-generated policies.

community, where investigations have continued under two main headings: *norms* and *policies*. Drawing on precedents in legal theory, social psychology, social philosophy, sociology, and decision theory [119], *norm-based* approaches have grown in popularity [11; 40; 75; 76]. In the multi-agent system research community, Conte and Castenfranchi [39] found that norms were variously described as constraints on behavior, ends or goals, or obligations. For the most part, implementations of norms in multi-agent systems share three basic features:

1. they are designed offline or
2. they are learned, adopted, and refined through the purposeful deliberation of each agent; and
3. they are enforced by means of incentives and sanctions.

Interest in *policy-based* approaches to multi-agent and distributed systems has also grown considerably in recent years (http://www.policy-workshop.org). While sharing much in common with norm-based approaches, policy-based perspectives differ in subtle ways. Whereas in everyday English the term *norm* denotes a practice, procedure, or custom regarded as typical or widespread, a *policy* is defined by the American Heritage Online dictionary as a "course of action, guiding principle, or procedure considered expedient, prudent, or advantageous." Thus, in contrast to the relatively descriptive basis and self-chosen adoption (or rejection) of norms, policies tend to be seen as prescriptive and externally-imposed entities. Whereas norms in everyday life emerge gradually from group conventions and recurrent patterns of interaction, policies are consciously designed and put into and out of force at arbitrary times by virtue of explicitly-recognized authority.[7] These differences are generally reflected in the way most policy-based approaches differ from norm-based ones with respect to the three features mentioned above. Policy-based approaches:

1. support dynamic runtime policy changes, and not merely static configurations determined in advance;
2. work involuntarily with respect to the agents, that is, without requiring the agents to consent or even be aware of the policies being enforced; thus aiming to guarantee that even the simplest agents can comply with policy; and
3. wherever possible they are enforced preemptively, preventing buggy or malicious agents from doing harm in advance, rather than rewarding them or imposing sanctions them after the fact.

In the following subsections, we define policy in the sense that it is used in this chapter (2.1) and distinguish it from related concepts (2.2). We then offer definitions of the two major types of policy (2.3), discuss both traditional focus areas and new challenges for policy management (2.4), and outline the most important aspects (2.5) and benefits (2.6).

[7]; While it is true that over time norms can be formalized into laws, policies are explicit and formal by their very nature at their origin.

2.1 What is policy?

In agent and distributed computing contexts, policy can be defined as *an enforceable, well-specified constraint on the performance of a machine-executable action by a subject in a given situation:*

- *enforceable:* In principle, an action controlled by policy must be of the sort that it can be prevented, monitored, or enabled by system infrastructure;
- *well-specified:* Policies are well-defined declarative descriptions;
- *constraint on the performance:* The objective of policy is to ensure, with or without the knowledge or cooperation of the entity being governed, that the policy administrator's intent is carried out with respect to whether the specified policy-governed action takes place;
- *machine-executable action:* In addition to purely machine-executable actions, we include situations where a person is responsible for completing an action and then somehow signaling that fact to the machine;
- *subject:* The subject is either a human or a hardware or software component—or a group of such entities;
- *situation:* Policy applicability may be determined by a variety of preconditions and contextual factors.

2.2 Distinguishing policy from related concepts

It is evident that not every constraint in an agent system should be managed as an element of policy. Nor should policy in the sense we are discussing it here be confused with other related concepts. For example, the term *policy* is often used to describe what we will call a "Big P" policy, referring to the sorts of high-level declarations of objectives or preferences that one finds in discussions of strategic policy, public policy, or foreign policy. While it is true that every policy of the sort we are concerned with (call them "little p") is motivated by some higher level objective, "Big P" policies comprise a diversity of elements, some of which involve real world considerations that go far beyond distributed computing issues. Resolving the ambiguities and contradictions of complex and "soft" goals, guidelines, and tradeoffs at the "Big P" level is more the stuff of human deliberation and automated planning than of policy management frameworks which are best suited to analysis and implementation of hard constraints *after* the difficult preliminary framing has been done.[8]

[8] As a matter of practice, while systems should be designed to accommodate policy changes by an authorized person or trusted component of the policy management infrastructure at any time, we do not normally allow agents themselves the same privilege—certainly not, at least, the agent on whom the policy is being enforced. This is consistent with the main premise of David Billings' desiderata for human-centered systems [10] ("Humans are responsible for outcomes in human-machine systems") which implicitly assumes a fundamental asymmetry between humans and today's agents. Notwithstanding this assumption, we expect the broad balance between human and agent initiative and responsibility to co-evolve commensurate with the degree of trust humans are willing (or required) to exercise in particular kinds of technology for specific contexts of use. Already people rely routinely on technology to do things automatically for them that were unthinkable not too long ago.

Policy management also should not be confused with planning or workflow management, which are related but separate functions. Planning mechanisms are generally *deliberative* (i.e., they reason deeply and actively about activities in support of complex goals) whereas policy mechanisms tend to be *reactive* (i.e., concerned with simple actions triggered by some environmental event) [53, pp. 161-162]. Whereas plans are a unified roadmap for accomplishing some coherent set of objectives, bodies of policy collected to govern some sphere of activity are made up of diverse constraints imposed by multiple potentially-disjoint stakeholders and enforced by mechanisms that are more or less independent from the ones directly involved in planning. Plans tend to be strategic and comprehensive, while policies, in our sense, are by nature tactical and piecemeal. In short, we might say that while policies constitute the "rules of the road"—providing the stop signs, speed limits, and lane markers that serve to coordinate traffic and minimize mishaps—they are not sufficient to address the problem of "route planning."

Policies should not be mistaken for business rules, for while their motivations sometimes overlap with policy-based approaches, these two different attempts to enforce regularities on complex systems have usually maintained a different focus. Analogously to the world of policies, we can distinguish between "Big B" and "little b" business rules. A "Big B" business rule "pertains to any of the constraints that apply to the behavior of people in the enterprise, from restrictions on smoking to procedures for filling out a purchase order" [70, p. 5]. On the other hand, "little b" business rules pertain "to the facts which are recorded as data and constraints on changes to the values of those facts. That is, the concern is what data may or may not be recorded in the information system" [70, p. 5]. Like "Big P" policies, "Big B" business rules have a much broader scope than "little p" policies. The "little b" rules, on the other hand, are certainly narrower than "little p" policies to the extent that the former are restricted to governing to the kinds of actions that can be performed on a particular instance of a business database rather than to a broader concept of action in general.

Finally, it should be realized that unwanted circumstances cannot be prevented, nor required events made to happen, by policy management mechanisms alone. A variety of potential failures must be considered and counteracted in the design of safe and effective agent systems including: extreme events; hardware failure; human error; incorrect system design, specification, or implementation; and inconsistency, redundancy, inaccuracy, or incompleteness of agent knowledge and system information [53, pp. 245-246].

2.3 Types of policy

Drawing on their long history of policy research, Sloman and his colleagues define the two major types of policy, *authorizations* and *obligations:*

- "A positive authorization policy defines the actions that a subject is permitted to perform on a target. A negative authorization policy specifies the actions that a subject is forbidden to perform on a target" [41].
- "Obligation policies specify the action that a subject must perform on a set of target objects when an event occurs. Obligation policies are always triggered by events, since

the subject[9] must know when to perform the specified action" [41].[10]

2.4 Scope of policy management

The scope of policy management includes traditional focus areas such as security (e.g., confidentiality, availability, integrity, accountability, auditability, access control, intrusion detection[11]), resource management (e.g., controls over rate and amount of CPU, memory, and network consumption; Quality of Service guarantees), and information sharing and dissemination, but also goes beyond these in significant ways. For example, KAoS pioneered the concept of agent conversation policies in the mid-1990s [16; 17; 57; 58] and, in conjunction with Nomads, has explored the uses of mobility policies for several years [69]. Additional new challenges for policy management include:

- Sources and methods protection, digital rights management, information filtering and transformation, capability-based access;
- Active networks, agile computing, pervasive and mobile systems;
- Organizational modeling, coalition formation, formalizing cross-organizational agreements;
- Trust models, trust management, information pedigrees;
- Effective human-machine interaction: interruption/notification management, presence management, adjustable autonomy, teamwork facilitation, safety; and
- Intelligent retrieval of all policies relevant to some situation.

2.5 Aspects of policy management

Effective policy management involves many aspects including:

- policy negotiation;

[9] In the KAoS policy management framework (see Section 3.2), a type of enforcer called an *enabler* can be defined to assist subjects in fulfilling obligations, thus reducing, or ideally eliminating, the need for the agent itself to fully understand the policy and know when how to undertake its responsibilities [20; 117]. Enablers can also be defined for some types of authorization policies (see Section 3.2.6).

[10] Some systems differentiate a second class of obligations that requires a given desired state to be continuously maintained by an unspecified action (e.g., Agent A must maintain at least 10 widgets in the bin) in contrast to normal obligations that require a specific action to be performed in response to a trigger (e.g., IF the number of widgets $<= 10$ THEN Agent must fill the bin with widgets). For example, Pynadeth and Tambe [91] distinguish between four classes of *safety constraints:* forbidden actions, forbidden states, required actions, and required states. In KAoS (see Section 3.2), forbidden actions correspond to negative authorization policies, while required actions and states map to positive obligation policies. Since many states of the world are outside of system control and cannot be forbidden *a priori*, they can best be handled by representing a forbidden state (ideally with some safety margin) as a trigger to a positive obligation policy that requires the agent to achieve a permissible state.

[11] Policies provide a mechanism for organizations to insert and activate their policies (and thus their intentions) right into the beating heart of modern distributed systems.

- analysis (e.g., multi-organizational policy integration, conflict detection and resolution, enforceability analysis, “what if” analyses and simulations, validation);
- persistence and reuse;
- distribution;
- disclosure;
- monitoring and logging, compliance detection, and enforcement;
- precedent management (tracking circumstances where policies have been overridden in the past);
- visualization; and
- verification.

Fortunately, the development of a comprehensive policy management approach does not require a duplication of the extensive investment that has been made in research on security mechanisms over many years. Instead, policy management capabilities can be made to leverage and extend this ongoing research as they incorporate appropriate security capabilities into various aspects of policy enforcement and into mechanisms for the protection of the policy management components themselves. Security policies represent only a (well-studied) subset of the many interacting types of policy.

2.6 Benefits of policy management

A policy-based approach has many benefits:

Explicit license for autonomous behavior. Policy representations that the description of entities and actions at abstract levels (e.g., ontologies) can beneficially underspecify the constraints of policy, giving human stakeholders as much leeway as they require to shape the limits of agent behavior across an arbitrarily large scope of action, while leaving every unmentioned detail completely in the hands of the agents that are closest to the problem. Thus, the coupling of policy with autonomy enables human organizations to *think globally while acting locally*. In short, rather than mistakenly thinking of policy only as a restrictive nuisance, we might more productively think of it as the explicit license by which agents are authorized to make specific decisions and adaptations autonomously in response to novel problems and opportunities as they arise—without violating the constraints imposed by those who are responsible for their behavior.

Reusability. Policies encode sets of useful constraints on agent or component behavior, packaging them in a form in which they can easily be reused as occasions require. By reusing policies when they apply, we reap the lessons learned from previous analysis and experience while saving the time it would have taken to reinvent them from scratch. Policy libraries can package sets of policies that have been pre-approved for particular situations. For example, military applications may have defined different policy sets that come into play for various levels of threat conditions.

Efficiency. In addition to lightening the application developers' workload, well-defined policy management mechanisms can sometimes increase runtime efficiency [91]. For example, to the extent that policy conflict resolution can be performed offline in advance and policies can be converted to an efficient runtime representation, overall performance can be increased [20; 117].

Extensibility. A well-designed policy management capability provides a layer of basic representations and services which can be straightforwardly extended to diverse and evolving platforms and sets of operational capabilities that are often subject to rapid rates of technology refresh. Ideally, these modifications could be made without extensive manual markup or duplication of information stored elsewhere in the organization.

Context-sensitivity. Explicit policy representation improves the ability of agents, components, and platforms to be responsive to changing conditions without changing their code. In mature policy management systems, such changes to policy can be made manually through convenient distributed administration capabilities or triggered programmatically by events.

Verifiability. By representing policies in an explicitly declarative form instead of burying them in the implementation code, we can better support important types of policy analysis [53, pp. 156-157]. First—and this is absolutely critical for security policies—we can externally validate that the policies are sufficient for the application's tasks, and we can bring both automated theorem-provers and human expertise to this task. Second, there are methods to ensure that agent behavior which follows the policy will also satisfy many of the important properties of reactive systems: liveness, recurrence, safety invariants, and so forth.

Support for simple as well as sophisticated agents. By putting the burden for policy analysis and enforcement on the infrastructure rather than having to build such knowledge into each of the agents themselves, we ensure that all agents operate within the bounds of policy constraints [19]. In this way, even one agent shall not be lost due to policy violations, no matter how simple or sophisticated the agent's design, and the task of agent developers is thereby reduced in complexity [58].

Protection from poorly-designed, buggy, or malicious agents. Intelligent systems functioning in complex environments cannot rely on design-time techniques to completely eliminate the possibility of unwanted events occurring during operations.[12] Moreover, even if it could be guaranteed that agents designed by a given group would always function correctly, the fact is that so long as reliance on open systems continues to increase the possibility of buggy or malicious agents designed by others cannot be completely ignored. Various forms of policy-based barriers that can control the actions of such agents through monitoring, analysis, inference, adjustable autonomy, and enforcement methods that are infrastructure-based and independent of the agents' own reasoning appear to be the most effective way to reduce the risk of these serious problems [71, pp. 414-431].

Reasoning about agent behavior. As permitted by policy disclosure policies [98], sophisticated agents can reason about the implications of the policies that govern their behavior and the behavior of other agents. To the extent that behavior can be predicted from policy, making accurate and consistent models of agents becomes more feasible.

[12] As Fox and Das [53, p. 158] wisely observe, "the nature of a hazard will frequently be unknown until it actually arrives. In some circumstances ensuring that a system reliably does what the designers intended—and only what they intended—may be exactly the wrong thing to do!"

3 Technical Aspects of Agent Acceptability

Examples of the kinds of basic infrastructure that will be required to support the technical aspects of agent acceptability are becoming more available. Designed from the ground up to exploit next-generation Internet and Web-Services capabilities, grid-based approaches, for example, aim to provide a universal source of dynamically pluggable, pervasive, and dependable computing power, while guaranteeing levels of security and quality of service that will make new classes of applications possible ([52]; http://www.gridforum.org). By the time these sorts of approaches become mainstream for large-scale applications, they will also have migrated to ad hoc local networks of very small devices [55; 109].

This being said, however, we must go far beyond these current efforts to enable the vision of long-lived agent communities performing critical tasks (Figure 7). Current infrastructure implementations typically provide only very simple forms of resource guarantees and no incentives for agents and other components to look beyond their own selfish interests. At a minimum, future infrastructure must go beyond the bare essentials to provide pervasive *life support services* (relying on mechanisms such as orthogonal persistence and strong mobility [105; 106]) that help ensure the survival of agents that are designed to live for many years. Beyond the basics of individual agent protection, long-lived agent communities will depend on *legal services*, based on explicit policies, to ensure that rights and obligations are monitored and enforced. Benevolent *social services* might also be provided to proactively avoid problems and help agents fulfill their obligations. Although some of these elements exist in embryo within specific agent systems, their scope and effectiveness has been limited by the lack of underlying support at both the platform and application levels.

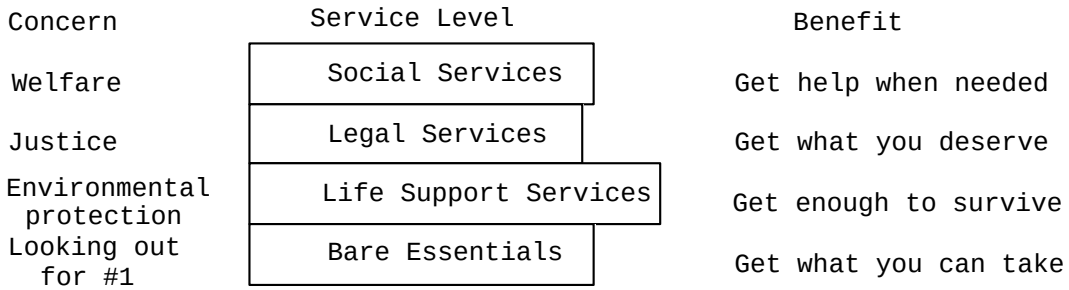

Figure 7: Required elements of future infrastructure for software agents.

In this section we describe how we are working toward extending current agent infrastructure to provide support for rudimentary versions of these kinds of agent services. We will first describe Nomads life support services (3.1). Then we will show how we are exploring the basics of legal and social services to agents through the use of KAoS domain and policy management models and mechanisms (3.2). Then we briefly describe several applications of Nomads and KAoS (3.3) and give examples of policy types relating to the technical aspects of agent acceptability (3.4).

3.1 Nomads Life Support Services

Nomads is the name we have given to the combination of Aroma, an enhanced Java-compatible Virtual Machine (VM), with its Oasis agent execution environment [105; 106]. In its current

version, it is designed to provide basic *life support services* ensuring agent environmental protection of two kinds:

- assurance of availability of system resources, even in the face of buggy agents or deliberate denial-of- service attacks (3.1.1); and
- protection of agent execution state, even in the face of unanticipated system failure (3.1.2).

3.1.1 Protection of agent resources.

Our approach for life support services has thus far been two-pronged: enabling as much protection as possible in standard Java VMs while also providing Nomads and the enhanced Aroma VM for those agent applications that require it.

Although Java is currently the most popular and arguably the most mobility-minded and security-conscious mainstream language for agent development, current versions fail to address many of the unique challenges posed by agent software. While few if any requirements for Java mobility, security, and resource management are entirely unique to agent software, typical approaches used in non-agent software are usually hard-coded and do not allow the degree of on-demand responsiveness, configurability, extensibility, and fine-grained control required by agent-based systems.

For agents running in the Aroma VM, we can create a guarded environment that is considerably more powerful in that it not only provides the standard Java enforcement capabilities, but also supports access revocation under all circumstances, dynamic resource control and full state capture on demand for any Java agent or service.

To fully appreciate the resource control features of Aroma and Nomads, some understanding of the current Java security model is needed. Early versions of Java relied on the sandbox model to protect mobile code from accessing dangerous methods. In contrast, the security model in the current Java 2 release is permission-based. Unlike the original "all or nothing" approach, Java applets and applications can be given varying amounts of access to system resources. Unfortunately, current Java mechanisms do not address the problem of resource control. For example, while it may be possible to prevent a Java program from writing to any directory except /tmp (an access control issue), once the program is given permission to write to the /tmp directory, no further restrictions are placed on the program's I/O (a resource control issue). As another example, there is no way in the current Java implementation to limit the amount of disk space the program may use or to control the rate at which the program is allowed to read and write from the network.

Resource control is important for several reasons. First, without resource control, systems and networks are open to denial of service attacks through resource overuse. Second, resource control lays the foundation for quality-of-service guarantees. Before any quality-of-service guarantees can be made about the availability of resources, the system must be able to limit resource utilization of other tasks (which is currently not possible in the Java environment). Third, resource control presupposes resource accounting, which allows the resources consumed by some component of a system (or the overall system) to be measured for either billing or monitoring purposes. Monitoring resource utilization over time allows the detection of abnormal behavior as part of the system.

Finally, the availability of resource control mechanisms in the environment simplifies the task of developing systems for resource-constrained situations. Consider the task of developing and deploying a new system requiring concurrent execution and resource sharing with

existing systems. In such scenarios, the developer of the new system often has to limit the resource utilization of the new system in order to not interfere with the operations of the existing systems (for example, maybe the new system can only use 500 Kb/sec of network bandwidth because the rest of the available network bandwidth is required by the existing systems). Providing such a guarantee requires significant effort on behalf of the developer of the new system. However, if the underlying environment were to provide resource control mechanisms, then the new system could simply make a request to the underlying environment, which can then provide the necessary guarantees.

Aroma currently provides a comprehensive set of resource controls for CPU, disk, and network usage (Figure 8). The resource control mechanisms allow limits to be placed on both the rate and quantity of resources used by Java threads. Rate limits include CPU usage, disk read rate, disk write rate, network read rate and network write rate. Rate limits for I/O are specified in bytes/millisecond. Quantity limits include disk space, total bytes written to disk, total bytes read from the disk, total bytes written to the network, and total bytes read from the network. Quantity limits are specified in bytes. One of the major benefits of the Aroma VM is that resource controls are transparent to the Java code executing inside the VM. In particular, the enforcement of the resource limits does not require any modifications to the Java code. Also, the existence of rate limits (and their enforcement) is completely transparent to the Java component or service.

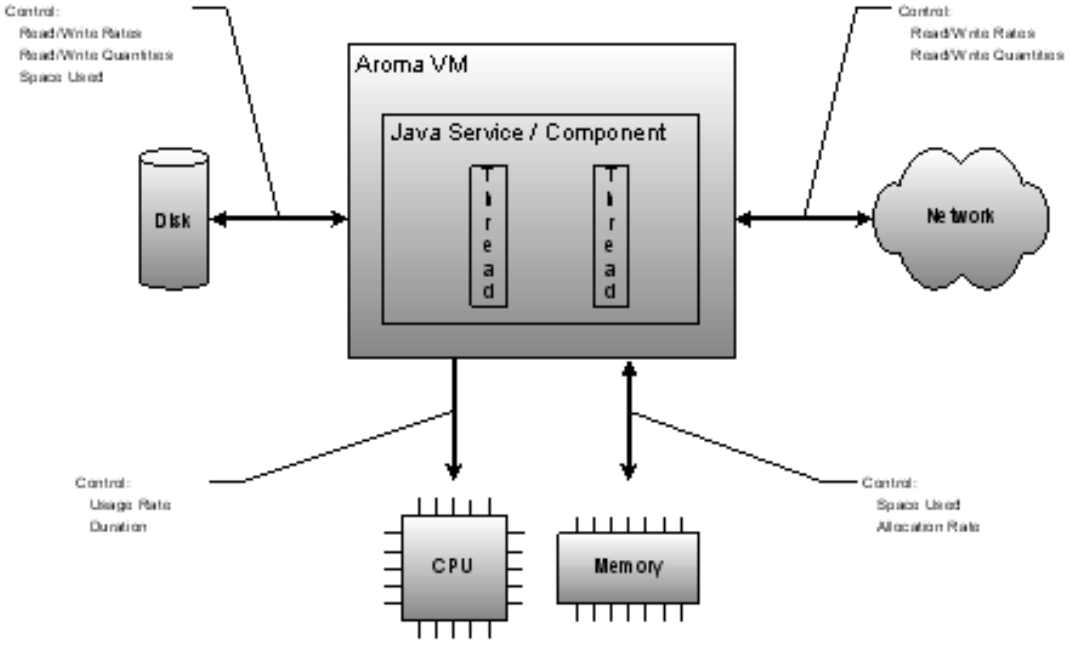

Figure 8: The Aroma VM provides resource control for CPU, memory, disk, and network.

CPU resource control was designed to support two alternative means of expressing the resource limits. The first alternative is to express the limit in terms of bytecodes executed per millisecond. The advantage of expressing a limit in terms of bytecodes per unit time is that given the processing requirements of a thread, the thread's execution time (or time to complete a task) may be predicted. Another advantage of expressing limits in terms of bytecodes per unit time is that the limit is system and architecture independent. The second alternative is to express the limit in terms of some percentage of CPU time, expressed as a

number between 0 and 100. Expressing limits as a percentage of overall CPU time on a host provides better control over resource consumption on that particular host.

Rate limits for disk and network are expressed in terms of bytes read or written per millisecond. If a rate limit is in effect, then I/O operations are transparently delayed if necessary until such time that allowing the operation would not exceed the limit. Threads performing I/O operations will not be aware of any resource limits in place unless they choose to query the VM.

Quantity limits for disk and network are expressed in terms of bytes. If a quantity limit is in effect, then the VM throws an exception when a thread requests an I/O operation that would result in the limit being exceeded.

In agent environments, several uses of the Nomads-based resource control mechanisms are possible. First, the KAoS domain and policy services (see Section 3.2) will be able to utilize the resource control capabilities in order to place limits on the resources consumed by services and components running within the Aroma VM. The KAoS Guard will be able to vary the resource limits to accommodate changes in policy or level of service guarantees. The Guard will also be able to take advantage of the resource accounting capabilities to measure and report back on the resources consumed by services and components and, if policy permits, to look for patterns of resource abuse that might signal denial-of-service attacks and take autonomous action to reduce resources to the attacker accordingly.

We are working with Sun Microsystems Laboratories to help incorporate resource control capabilities into commercial Java Virtual Machines. Incorporating Aroma-like resource control mechanisms into Java will enable agents and other distributed systems applications to run in more secure environments.

3.1.2 Protection of agent state

With respect to protection of agent state, we need a way to save the entire state of the running agent or component, including its execution stack, at anytime so it can be fully restored in case of system failure or a need to temporarily suspend its execution. The standard term describing this process is checkpointing. Over the last few years, the more general concept of transparent persistence (sometimes called “orthogonal persistence”) has also been developed by researchers at Sun Microsystems and elsewhere [65]. The goal of this research is to define language-independent principles and language-specific mechanisms by which persistence can be made available for all data, irrespective of type. Ideally, the approach would not require any special work by the programmer (e.g., implementing serialization methods in Java or using transaction interfaces in conjunction with object databases), and there would be no distinction made between short-lived and long-lived data.

The Aroma VM has been enhanced with the capability to capture the execution state of any running Java program. Current commercial Java VMs do not provide any mechanisms to capture the execution state of a Java program. The state capture mechanism is used to provide strong mobility for Nomads agents, which allows them to request mobility no matter at what point they are in running their code. Without strong mobility, the code for a mobile agent needs to be structured in a special way to accommodate migration operations. Strong mobility allows agents to be mobile without any special structural requirements (see discussion of resource redirection below).

Promising work has been done on translating agents that use strong mobility into agents that use weak mobility. These approaches work well for agents that are single threaded and

do not require asynchronous state capture.[13] The Aroma VM supports capturing of execution state of both multi-threaded agents and allows external events to trigger state capture operations.

We have used the state capture features of Nomads extensively for agents requiring anytime mobility, whether in the performance of some task or for immediate escape from a host under attack or about to go down (we call this scenario "scram"). We have also put these features to use for transparent load-balancing and forced code migration on demand in distributed computing applications [112]. To support transparent persistence for agents and agent infrastructure components, we are implementing scheduled and on-demand checkpointing services that will protect agent execution state, even in the face of unanticipated system failure.

Forced migration of agents would fail if the agent were using local resources (such as files or network endpoints) on the original host. To solve this problem, we have also implemented transparent redirection of resource access for files and TCP sockets. For example, network redirection is provided through a mechanism called Mockets (mobile sockets) [82], which allow an agent to keep a TCP socket connection open while moving from host to host. Resource redirection is an important requirement to achieve full forced migration of agents.

3.2 KAoS Legal and Social Services

The technical aspects of agent acceptability involve other concerns besides the regulation of computing resources and protection of agent state. As the scale and sophistication of agents grow, and their lifespan becomes longer, agent developers and users will want the ability to express complex high-level constraints on agent behavior within a given environment. Within KAoS, these constraints are provided by policy.

Section 3.2.1 gives an overview of KAoS domain and policy services, which are intended to address this requirement. KAoS services rely on the KAoS Policy Ontologies (KPO), a set of description-logic-based ontologies of the computational environment, application context, and the policies themselves (3.2.2). Specification of domains and policies is performed through KPAT, the KAoS Policy Administration Tool (3.2.3). As new policies or other changes come into force, policy conflict resolution inference procedures determine which policies are in conflict and how to resolve these conflicts (3.2.4). KAoS policy distribution and disclosure mechanisms are described in Section 3.2.5. Finally, KAoS provides various enforcement mechanisms (3.2.6). Generic enforcer interfaces, with implementations for various platforms and action types, are responsible for prevention of or warnings about actions inconsistent with current policy. Similarly, generic enablers are responsible for performing, facilitating, or monitoring of required actions.

3.2.1 KAoS overview

KAoS is a collection of componentized agent services compatible with several popular agent frameworks, including Nomads [110], the DARPA CoABS Grid [67], the DARPA ALP/Ultra* Log Cougaar framework (http://www.cougaar.net), CORBA (http://www.omg.org), and Voyager (http://www.recursionsw.com/osi.asp). The adaptability of KAoS is due in large part

[13] By asynchronous, we mean a request to capture state that is generated by an external unexpected event or interrupt.

to its pluggable infrastructure based on Sun's Java Agent Services (JAS) (http://www.java-agent.org). For additional descriptions and perspectives on KAoS, the reader is referred to [16; 17; 19; 20]. While initially oriented to the dynamic and complex requirements of software agent applications, the services are also being adapted to general-purpose grid computing (http://www.gridforum.org) and Web services (http://www.w3.org/2002/ws/) environments as well.

KAoS domain services provide the capability for groups of agents to be structured into organizations of agent domains and subdomains to facilitate agent-agent collaboration and external policy administration. Domains may represent any sort of group imaginable, from potentially complex organizational structures to administrative units to dynamic task-oriented teams with continually changing membership. A given domain can extend across host boundaries and, conversely, multiple domains can exist concurrently on the same host. Domains may be nested indefinitely and, depending on whether policy allows, agents may become members of more than one domain at a time.

KAoS policy services allow for the specification, management, conflict resolution, and enforcement of policies within domains. Policies are represented in DAML (DARPA Agent Markup Language) as ontologies. The KAoS Policy Ontologies (KPO) distinguish between *authorizations* (i.e., constraints that permit or forbid some action) and *obligations* (i.e., constraints that require some action to be performed, or else serve to waive such a requirement) [41]. Through various property restrictions in the action type, a given policy can be variously scoped, for example, either to individual agents, to agents of a given class, to agents belonging to a particular group, or to agents running in a given physical place or computational environment (e.g., host, VM).

Some of the important features of KAoS are worth noting. First, the approach does not assume that we are dealing with a homogeneous set of agents that have been designed in advance to work with KAoS services. Rather the goal is to be able to have KAoS services work with arbitrarily written agents after the fact through support being added transparently at the platform level. Second, insofar as possible the KAoS framework supports dynamic runtime policy changes, and not merely static configurations determined in advance. Third, the framework is extensible to a variety of execution platforms that might be simultaneously running with different enforcement mechanisms—initially agent platforms implemented in Java and Aroma [105; 106]—but in principle any platform for which policy enforcement mechanisms may be written. For example, we are now extending KAoS to work in conjunction with the Globus Toolkit version 3. Fourth, the KAoS framework is intended to be robust and adaptable in continuing to manage and enforce policy in the face of attack or failure of any combination of components. Finally, we address the need for easy-to-use policy-based administration tools capable of containing domain knowledge and conceptual abstractions that let application designers focus their attention more on high-level policy intent than on implementation details. Such tools require sophisticated graphical user interfaces for monitoring, visualizing, and dynamically modifying policies at runtime.

3.2.2 KAoS Policy Ontologies (KPO)

The representation chosen to describe the policies and their context largely determines the flexibility, extensibility, and amenability to analysis of a given implementation. KAoS services rely on a DAML description-logic-based ontology of the computational environment, application context, and the policies themselves that enables runtime extensibility and adapt-

ability of the system, as well as the ability to analyze policies relating to entities described at different levels of abstraction. The representation facilitates careful reasoning about policy disclosure, conflict detection, and harmonization, and about domain structure and concepts. The representation of classes in ontologies means that the effect of policies can be extended automatically through subsumption reasoning to new classes of objects defined at a later time.

Designed to support the emerging "Semantic Web," DAML is the latest in a succession of Web markup languages (http://www.daml.org; [9]). HTML, the first Web markup language, allowed users to markup documents with a fixed set of formatting tags for human use and readability. XML allows users to add arbitrary structures to their documents but expresses very little directly about what the structures mean. RDF (Resource Description Format) encodes meaning in sets of subject-verb-object triples, where elements of these triples may each be identified by a URI (typically a URL).

DAML extends RDF to allow users to specify ontologies composed of classes and properties, as well as inference rules. The ontologies can be used by people for a variety of purposes, such as enabling more accurate or complex Web searches. Agents can also use semantic markup languages to understand and manipulate Web content in significant ways; to discover, communicate, and cooperate with other agents and services; or, as we outline in this chapter, to interact with policy-based management services and control mechanisms. OWL (Ontology Web Language), a W3C-approved evolution of DAML, is nearing final release (http://www.w3.org/2001/sw/).

The current version of KPO defines basic ontologies for actions, actors, groups, places, various entities related to actions (e.g., computing resources), and policies. There are currently 79 classes and 41 properties defined in the basic ontologies. It is expected that for a given application, the ontologies will be further extended with additional classes, individuals, and rules.

Actors and groups. Groups of actors or other entities may be distinguished according to whether the set of members is defined extensionally (i.e., through explicit enumeration in some kind of registry) or intensionally (i.e., by virtue of some common property such as a joint goal that all actors possess or a given place where various entities may temporarily or permanently be located).

Policies. In KAoS, a policy is represented as a DAML instance of the appropriate policy type with associated values for properties: priority, update time stamp and a site of enforcement.[14] The most imported property value is the name of a controlled action class. In most cases a new action class is built automatically whenever a policy is defined. Through various property restrictions, a given policy can be variously scoped, for example, either to individual agents, to agents of a given class, to agents belonging to a particular group, or to agents running in a given physical place or computational environment (e.g., host, VM). Additional aspects of the action context can be precisely described by restricting values of its properties.

The policy example below stipulates that the members of a domain named Arabello-HQ are forbidden to communicate with those outside this domain using unencrypted communication. The syntax of this example may seem very complex, however, the DAML policy is not meant to be written or analyzed directly by a human. Note that the use of the KPAT user interface would hide the complexity of this representation from the policy administrator.

[14]While we realize that the use of DAML (and the more powerful OWL in the future) restricts the representation of policy, it serves to make inference considerably more tractable. The classic tradeoffs between representational expressiveness, tractability of inference, and human understandability are explored in [72] and [48].

```
< daml : Classrdf : ID = "PlAction" >
  < rdfs : subClassOfrdf : resource = "#CommunicationAction"/ >
    < rdfs : subClassOf >
      < daml : Restriction >
        < daml : onPropertyrdf : resource = "#performedBy"/ >
        < daml : toClassrdf : resource = "#MembersOfDomainArabello − HQ"/ >
      < daml : Restriction >
    < rdfs : subClassOf >
    < rdfs : subClassOf >
      < daml : Restriction >
        < daml : onProperty   rdf : resource = "#hasDestination"/ >
      < daml : toClassrdf : resource = "#notMembersOfDomainArabello − HQ"/ >
    < /daml : Restriction >
  < /rdfs : subClassOf >
< /daml : Class >
< policy : NegAuthorizationPolicyrdf : ID = "P1" >
  < policy : controlsrdf : resource = "#P1Action"/ >
  < policy : hasSiteOfEnforcementrdf : resource = "#ActorSite"/ >
  < policy : hasPriority > 1 < /policy : hasPriority >
  < policy : hasUpdateTimeStamp > 446744445544 < /policy : hasUpdateTimeStamp >
< /policy : NegAuthorizationPolicy
```

3.2.3 Domain and policy specification using KPAT

The basic policytool that Java currently provides assists users in editing Java policy files. However, to be useful and usable in realistic settings, policy-based administration tools should contain domain knowledge and conceptual abstractions to allow applications designers to focus their attention more on high-level policy intent than on the details of implementation. Moreover, while Java provides only for static policies, critical agent applications will require tools for the monitoring, visualization, and dynamic modification of policies at runtime.

The KAoS Policy Administration Tool (KPAT[15]) implements a graphical user interface to policy and domain management functionality. It has been developed to make policy management easier for administrators without requiring extensive training (Figure 9). Using KPAT, an authorized user may make changes to agent policy from anywhere using a secure Web browser. Alternatively, trusted infrastructure components such as Guards (see Section 3.2.5) may, if authorized, propose policy changes autonomously or semi-autonomously based on their observation of system events.

KPAT can be used to browse and load ontologies, to define, deconflict, and commit new policies, and to modify or delete old ones. Groups of interdependent policies can be composed into *policy sets*. The generic DAML Policy Editor is a powerful view that allows administrators fine-grained control over any aspect of policy specification. It is driven by the ontologies loaded into the Java Theorem Prover (JTP-see Section 3.2.4), and its constraint-driven interface always provides the user with the list of choices narrowed to only those appropriate to the context of the other current selections. Custom editors tailored to particular kinds of policies may also be added to KPAT and will be automatically invoked by default if a policy about the action class associated with the given custom editor is selected for editing.

[15]Pronounced KAY-pat.

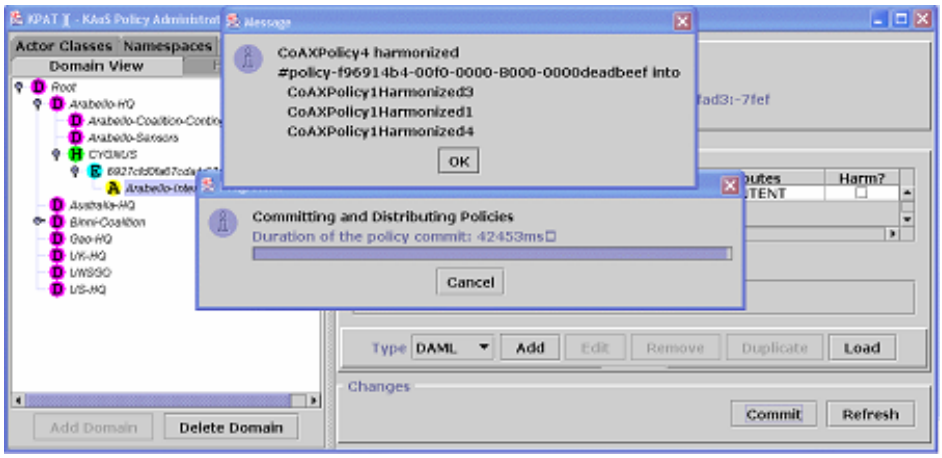

Figure 9: KPAT shown notifying the user of the results of policy harmonization.

When a user commits a change to an ontology (e.g., a new or edited policy, or changes to a domain structure) the Jena framework (http://www.hpl.hp.com/semweb/) is used to dynamically build a DAML representation based on the values selected by the user.

3.2.4 Policy conflict resolution

The KAoS Policy Ontologies are intended for a variety of purposes. One obvious application is during inference relating to various forms of online or offline analysis. They can be used for a variety of purposes, including policy disclosure management, reasoning about future actions based on knowledge of policies in force, and in assisting users of policy specification tools to understand the implications of defining new policies given the current context and the set of policies already in force.

Changes or additions to policies in force, or a change in status of an actor (e.g., a human administrator being given new permissions; an agent joining a new domain or moving to a new host) requires logical inference to determine, first of all, which policies are in conflict and, second, how to resolve these conflicts [79].

Figure 10 shows the three types of conflict that can currently be handled: positive vs. negative authorization (i.e., being simultaneously permitted and forbidden from performing some action), positive vs. negative obligation (i.e., being both required and not required to perform some action), and positive obligation vs. negative authorization (i.e., being required to perform a forbidden action). We have developed policy deconfliction and harmonization algorithms within KAoS to allow policy conflicts to be detected and resolved even when the actors, actions, or targets of the policies are specified at vastly different levels of abstraction, and whose initial results promise a high degree of efficiency and scalability. These algorithms rely in part on a version of Stanford's Java Theorem Prover (JTP; http://www.ksl.stanford.edu/software/JTP/) that we have integrated with KAoS.

Recently, we have modified KAoS conflict resolution handling to obviate the need for harmonization in many cases, further increasing performance. In the future we will add additional algorithms to help users understand and predict situations where a change in policy may lead to unintended consequences.

Policy precedence conditions. Policy precedence conditions are needed to properly execute the automatic conflict resolution algorithm. When policy conflicts occur, these condi-

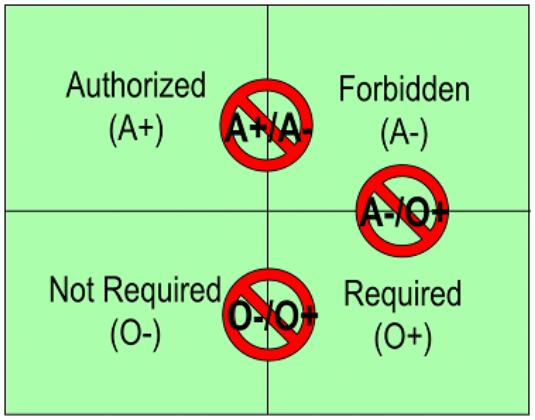

Figure 10: Three types of policy conflict.

tions are used to determine which of the two policies being compared is most important. The conflict can then be resolved automatically in favor of the most important policy. Alternatively, the conflicts can be brought to the attention of a human administrator who can make the decision manually.

We currently rely exclusively on the combination of numeric policy priorities[16] and update times to determine precedence—the larger the integer and the more recent the update the greater the priority. This is consistent with the natural intuition that more recent and higher priority policies should trump older and lower priority ones.

In the future we intend to allow people additional flexibility in designing the nature and scope of precedence conditions. For example, it would be possible to define precedence based on the relative authorities of the individual who defined or imposed the policies in conflict, which policy was defined first, which has the largest or smallest scope, whether negative or positive authorization trumps by default, whether subdomains takes precedence over superdomains or vice versa, and so forth.

Details of policy conflict resolution are explained in [20; 117].

3.2.5 Policy distribution and disclosure

Figure 11 shows the major components of the KAoS policy and domain services architecture. KAoS Domain Managers (DM) work together with JTP and the Policy Directory Service (PD) to ensure policy consistency at all levels of a domain hierarchy, to notify Guards about changes in policy or other aspects of system state that may affect their operation, and to handle persistence and queries to the PD. Because DM's are stateless, one DM instance may serve multiple domains or conversely, a single large domain may require several instances of the DM to achieve scalable performance.

Policies are stored within ontologies in the PD. Although DM's normally provide the limited public interface to the PD, private interfaces may allow the PD to be accessed by other authorized entities in accordance with policy disclosure strategies [98]. For example, trusted agents could be allowed to perform queries concerning domain policies in advance of submitting a registration request to a new domain. Because the policies in the directory

[16] In the absence of an explicitly rated priority, the priority value of a policy may be "inherited" from the person who defined the policy or the priority of the controlled action class.

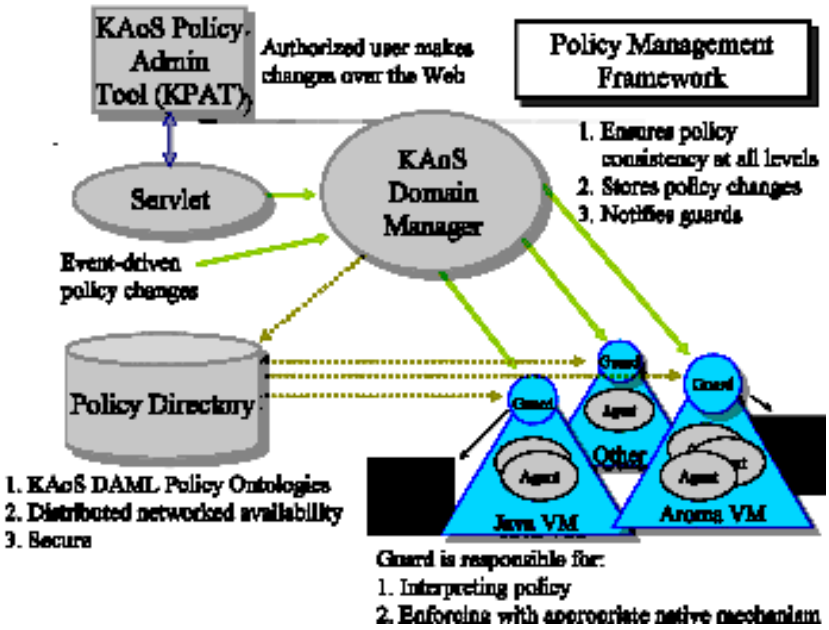

Figure 11: KAoS policy and domain services architecture.

service are expressed declaratively, some forms of analysis and verification can be performed in advance and offline, permitting execution mechanisms to be as efficient as possible.

Guards are responsible for policy enforcement within the bounds of a specified computational environment. They interpret policies that have been approved by the DM and enforce them with appropriate native enforcement mechanisms. While KPAT, DMs, and the Guards are intended to work identically across different agent platforms (e.g., DARPA CoABS Grid, Cougaar) and execution environments (e.g., Java VM, Aroma VM), enforcement mechanisms are necessarily designed for a specific platform and execution environment.

3.2.6 Policy enforcement

Types of enforcement mechanisms. While other components of KAoS policy and domain services are generic, enforcement mechanisms are necessarily platform specific. In applications to date, we have relied on several different kinds of enforcement mechanisms. Enforcement mechanisms built into the execution environment (e.g., OS or Virtual-Machine-level protection) are the most powerful sort, as they can generally be used to assure policy compliance for any agent or program running in that environment, regardless of how that agent or program was written. For example, the Java Authentication and Authorization Service (JAAS) provides methods that tie access control to authentication. In KAoS, we have in the past developed methods based on JAAS that allow policies to be scoped to individual agent instances rather than just to Java classes. Currently, JAAS can be used with Java VMs; in the future it should be possible to use JAAS with the Aroma VM as well. As described above, the Aroma VM provides, in addition to Java VM protections, a comprehensive set of resource controls for CPU, disk and network. The resource control mechanisms allow limits to be placed on both the rate and the quantity of resources used by Java threads. Guards running on the Aroma VM can use the resource control mechanisms to provide enhanced security (e.g., prevent or disable denial-of-service attacks), maintain quality of service for given agents, or give priority

to important tasks.

A second kind of enforcement mechanism takes the form of extensions to particular agent platform capabilities. Agents that participate in that platform are generally given more permissions to the degree they are able to make small adaptations to comply with policy enforcer interface requirements. For example, in applications using the DARPA CoABS Grid, we have defined a KAoSAgentRegistrationHelper to replace the default GridAgentRegistrationHelper. Grid agent developers need only replace the class reference in their code to participate in agent domains and be transparently and reliably governed by policies currently in force. On the other hand, agents that use the default GridAgentRegistrationHelper do not participate in domains, and as a result they are typically granted very limited permissions in their interactions with domain-enabled agents.

Finally, a third type of enforcement mechanism is necessary for obligation policies. Because obligations cannot be enforced through preventive mechanisms, enforcers can only monitor agent behavior and determine after the fact whether a policy has been followed. For example, if an agent is required by policy to report its status to its supervisor every five minutes, an enforcer might be deployed to watch whether this is in fact happens, and if not to either try to diagnose and fix the problem, or alternatively take appropriate sanctions against the agent (e.g., reduce permissions or publish the observed instance of noncompliance to an agent reputation service).

Two sorts of enforcers can be used for obligation policies: *monitors* and *enablers.* An example of a monitor type of enforcer is given in the previous paragraph. *Enablers* go beyond simple monitoring to proactively facilitate or perform the obligation on behalf of the agent. For example, a monitor might not only watch whether the agent described above reports every five minutes, but actively facilitate the fulfillment of its obligation by querying its status every five minutes and making the report to its supervisor on its behalf.[17]

Site of enforcement. Each policy has a property that defines the site of policy enforcement. For example, access control policies are typically enforced by a mechanism directly associated with the resource to be protected (i.e., the *target*). However in some cases, administrators may not have control over this resource and instead may require the policy to be enforced by a mechanism associated with the actor (i.e., the *subject*) or some other entity under their purview.[18]

Authorization mechanisms and default modalities. Before performing its function to allow or forbid a given action, the enforcer must obtain an answer to the question, "is a given action authorized or not?" KAoS allows the answer to this authorization question to be given in one of three ways: through the use of the Directory Service, some intermediary module, or the enforcer itself.

For reasons of performance and robustness, it is preferable to answer the authorization question locally in the enforcer or an intermediary module whenever possible. In this way enforcement can also continue even if the connection to the Policy Directory Service is interrupted. In such cases, the worst that could happen is that the PD will be temporarily unable to notify enforcers about policy-relevant changes to the ontologies, but at least the current policy will continue to be enforced.

Simple parameter-based representations of relevant policies are stored locally in each

[17] *Enablers* can also be used in conjunction with authorization policies, as in our work with encryption under the auspices of the DARPA Ultra*Log program (see example in 3.4.2 below).

[18] This same motivation led the designers of Ponder to develop *refrain* policies for subject-side enforcement [41].

enforcer and used in answering the authorization question. An action description is passed to the enforcer, which traverses its policy storage and checks to see if the given action instance is in the range of actions controlled by any of its policies.

If the authorization mechanism does not find any policy applicable to the action description passed to it, it answers the question consistent with whatever default authorization modality it has been given. Default authorization modalities are currently configured on a per domain basis. The defaults correspond to a *democracy,* where everything is permitted that is not explicitly forbidden, or a *tyranny,* where everything is forbidden that is not explicitly permitted.

Additional details of KAoS policy enforcement are described in [20; 117].

3.3 Nomads and KAoS Applications

KAoS and Nomads policy and domain services are being extended and evaluated in the context of several applications.

The DARPA CoABS-sponsored Coalition Operations Experiment (CoAX) (http:// www. aiai.ed.ac.uk /project/ coax/) [4; 107] is a large international cooperation that models military coalition operations and implements agent-based systems to mirror coalition structures, policies, and doctrines. CoAX aims to show that the agent-based computing paradigm offers a promising new approach to dealing with issues such as the interoperability of new and legacy systems, the implicit nature of coalition policies, security, and recovery from attack, system failure, or service withdrawal. KAoS provides mechanisms for overall management of coalition organizational structures represented as domains and policies, while Nomads provides strong mobility, resource management, and protection from denial-of-service attacks for untrusted agents that run in its environment.

Within the DARPA Ultra*Log program (http://www.ultralog.net) we are collaborating with Network Associates (NAI) to extend and apply KAoS policy and domain services to assure the scalability, robustness, and survivability of logistics functionality in the face of information warfare attacks or severely constrained or compromised computing and network resources.

As part of the Army Research Lab Advanced Decision Architectures Consortium, we have been investigating the use of KAoS and Nomads technologies to enable soldiers in the field to use agents from handheld devices to perform tasks such as dynamically tasking sensors and customizing information retrieval [108; 111]. Suri has developed an agile computing platform [109] that provides a foundation for this work. We have also commenced an investigation of requirements for policy-based information access and analysis within intelligence applications.

An application focused more on the social aspects of agent policy is within the NASA Cross-Enterprise and Intelligent Systems Programs, where we are investigating the integration of Brahms, an agent-based design toolkit that can be used to model and simulate realistic work situations in space [29; 100] with KAoS policy-based models and Nomads strong mobility and resource control capabilities to drive human-robotic teamwork and adjustable autonomy for highly-interactive autonomous systems such as the Personal Satellite Assistant (PSA). The PSA is a softball-sized flying robot that is being developed to operate onboard spacecraft in pressurized micro-gravity environments [1; 18; 54; 101]. The same approach is also being generalized for use in other testbeds, such as in conjunction with Johnson Space

Center's Robonaut [5; 23] and Mini-AERCam. The Office of Naval Research (ONR) is supporting research to extend this work on effective human-agent interaction to unmanned vehicles and other autonomous systems that involve close, continuous interaction with people. As one part of this research IHMC (Jerry Pratt, James Allen, Jeff Bradshaw, Niranjan Suri) and University of South Florida (USF) (Robin Murphy) are developing a new robotic platform with carangiform (fish-like) locomotion, specialized robotic behaviors for humanitarian demining, cooperative agents, agile computing, and mixed-initiative human control.

Under funding from DARPA's Augmented Cognition Program, we are taking the challenge of effective human-agent interaction one step further as we investigate whether a general policy-based approach to the development of cognitive prostheses can be formulated, in which human-agent teaming could be so natural and transparent that robotic and software agents could appear to function as direct extensions of human cognitive, kinetic, and sensory capabilities [49; 62] (see Section 4.4). We are also involved in preliminary explorations of space applications of this technology, for example the development of advanced space suits that integrate a variety of cognitive and robotic prostheses.

3.4 Examples of Policy Types Relating to Technical Aspects of Agent Acceptability

To better describe the nature of policy as it relates to the technical aspects of agent acceptability, we now discuss several examples. These examples are not intended to be comprehensive, only illustrative. Some of them are related to actual policies that we have used in various applications of KAoS; others reflect use cases we have anticipated but not yet implemented.

Norman suggests that the technical considerations include such things as ensuring robustness against technical failures, guarding against error and maliciousness, and protecting privacy [87]—we touch on each of these considerations in some way in this section. Later on in the chapter (Section 4.3) we present examples of policies relating to social aspects of agent behavior. Admittedly the distinction between the two kinds of examples is not always clearcut.

For clarity's sake, we will present example policies in ordinary English rather than in DAML. For brevity's sake, the policies will be presented in an incomplete abbreviated form. Each example is preceded by A+, A-, O+, or O- to indicate whether it is respectively a positive authorization, negative authorization, positive obligation, or negative obligation.[19] We will look at six categories of policy: authentication (3.4.1), data and resource access and protection (3.4.2), communication (3.4.3), resource control (3.4.4), monitoring and response (3.4.5), and mobility (3.4.6).

3.4.1 Authentication policies

O+: *IF KPAT is launched*

THEN that instance of KPAT is required to successfully complete a strong authentication process within time T

PRECEDENCE: A+: allowing use of this instance of KPAT by a user in an Administrative Assistant role

ELSE O+: this instance of KPAT must terminate.

[19] As this chapter goes to press, the representation and enforcement of obligation policies is still under development.

In this example, which is typical of some of the policies developed within our DARPA Ultra*Log research, the policy assures that strong authentication will be performed each time an effort is made to launch KPAT. Strong authentication is an abstract action that can represent any number of more specific strong authentication methods in the ontology that are available to the system. The authentication might be performed by KPAT itself or delegated to an enabler. One could argue that this policy should be hard wired into the code rather than represented explicitly. That, however, would reduce flexibility in ways that may not be desirable. For example, KPAT administrators at times may want to take this policy out of force in an emergency situation.

In DAML, we represent the precedence conditions as one or more policies. In this case, a negative authorization policy forbids any use of KPAT until the conditions of the obligation policy are fulfilled. Roles are represented straightforwardly as merely one kind of domain or group to which human or agent actors belong. It is recommended to use time or some more general state indicator as one of the conditions of obligation fulfillment to minimize the risk of the agent getting "stuck" indefinitely. Consequences of non-fulfillment of the obligation (the "ELSE" clause) are also represented as policies. In this case, KPAT is obliged to terminate if the obligation is not successfully fulfilled. In subsequent examples we will not always list the precedence, conditions of fulfillment, or consequences of non-fulfillment explicitly.

A-: *A user is forbidden from taking any action with account A*

IF the user has login_failure_count >= n and time since failure <= T

This negative authorization policy, again representative of our Ultra*Log work, deals with authentication failure. After a given number of login failures, the user is locked out of the account until some period of time elapses.

O+: *IF the space station crewmember has issued a voice command*

THEN the Personal Satellite Assistant (PSA) is required to authenticate the crewmember's voice within time T

PRECEDENCE: A-: PSA is forbidden to perform the action corresponding to the command

ELSE O+: PSA notifies crewmember appropriately

This example is drawn from our NASA human-agent teamwork research. Since authorization for some PSA action may depend on who commanded it, authentication of the crewmember's voice is required before the action is performed. The "ELSE" clause embeds a notification policy (see Section 4.3.2 below).

3.4.2 Data and resource access and protection policies

A-: *Agent X is forbidden from saving data that is unsigned and/or unencrypted*

This data protection policy example specifies that agent X must sign and encrypt all data that it saves. In our work with Ultra*Log, the encryption would be performed by an enabler. In other words, each time X saves data, the policy is enforced through the enabler transparently doing the proper sort of encryption on its behalf.

A-: *All actors in Role R are forbidden from performing any action on servlet S*

This resource access policy prevents any unauthorized use of Java servlet S by actors (i.e., agents or humans) who are in role R. As in a previous example, the power of abstract specification in the ontology is highlighted: note that this policy can be specified without having to know in advance the particular actions that can be performed on the servlet.[20] Additional Ultra*Log examples of this kind include policies governing actions such as Java JAR file verification, limiting access to private keys, and predicate-based access restriction to blackboard information.

A+: *Users in Role CA Administrator are permitted to perform the revoke certificate operation on the CA Service*

Users or agents in a given role or with a given privilege are here authorized to revoke certificates.

An important part of our current investigations on policy-based information access for intelligence applications concerns policy disclosure policies. These sorts of policies control the kinds of intelligent responses that can be given as part of queries about which policies are relevant to a given user's analysis or decision-making context. In a related application, an agent may want to know about the policies of a given domain before it registers to join. Policy disclosure policies would determine what kind of policy information could be given that agent without compromising confidentiality. We are drawing on the work of [98; 122] to develop more complex strategies for policy disclosure in a variety of circumstances.

3.4.3 Communication policies

Communication has proven to be the most important application of policy within our CoAX research [4]. Typically, the domains are configured to be in the "tyrannical" mode, blocking communication among different countries, organizations, or functional groups unless specifically allowed. For example, administrators from the fictional country of Arabello decided on the following restrictive default policy for the actors in their domain:

A-: *Agents in the Arabello domain are forbidden from sending messages to any agent outside the Arabello domain*

However administrators from the Arabello contingent wanted to enable the Arabello Intel agent to be able to send a subset of its reports to the coalition. They specified the following policy, which was assigned a higher precedence for policy conflict resolution purposes:

A+: *Arabello Intel Agent is permitted to send messages about enemy diesel submarines to any member of the Binni-Coalition domain (sharing messages about any other topic was still forbidden)*

Communication blocking based on message content as illustrated in this example is facilitated by the use of a custom editor within KPAT that allows the administrator to specify the kinds of messages that are to be permitted based on DAML-typing of various message fields [107].

[20] Capability-based access is a term used by Suri to describe an additional level of protection, where all of the details of service implementation are hidden from the client for confidentiality purposes.

A+: *MAD Sensor Agent is permitted to send reports with image resolution X:Y to any member of the Arabello domain*

As part of CoAX, as well as in follow-on Army research, we have also addressed requirements for filtering and transformation of data [107; 111]. For example the provider of a Magnetic Anomaly Detector (MAD) sensor was willing to share its reports with Arabello, but only on condition that the sensor's signal could be appropriately downgraded in order to prevent Arabello from knowing the full extent of the sensor's capabilities. The policy enforcer-enabler used in this application could be configured by policy to allow three different types of data transformation: a) changes in image resolution, b) changes in frame rate and c) introduction of time lags to prevent transmission of a real time video feed.

Many other types of policy-based transformations could be envisioned for sensor data feeds. A policy enforcer-enabler could, for instance, be implemented to hide sensitive targets or classified infrastructures from the image. This would be used to prevent the release of unnecessary details to the requesting agent by blurring or editing the image appropriately. Another example is an agent that reduces the precision of coordinate values embedded in message content. More generally, such filtering and transformation techniques can be used for sources and methods protection, and as part of the management of information pedigrees and digital rights protection.

In the Ultra*Log application, policies are required to block both sending and receiving of certain kinds of messages. The fact that KAoS policies can specify whether the site of enforcement is to be associated with the subject or the target is useful for this purpose: both the sending and the receiving can be blocked at either the subject or the target side as convenience dictates. *Policy templates* developed for Ultra*Log allow users to specify a composite set of multiple policies more simply as if it were a single policy. To take a simple example, the details of blocking of both sending and receiving messages are accomplished through a simple user interface that presents policy specification options in terms of the more general concept of "communication blocking." As additional examples of communication policies, Ultra*Log also requires that administrators be able to specify which cryptographic modes, transport types, and message formats are allowable in a given context. It also requires limits on message size and system resources in message delivery.

3.4.4 Resource control policies

Whereas resource access policies govern whether or not a resource is made available, resource control policies go a step further to control the amount and rate of resource usage (e.g., CPU, memory, network, hard disk, screen space). For example as part of one of the CoAX scenarios the country of Gao requests permission to host one of its agents on a sensor platform. Because its intentions are unclear and it is distrusted, it is required to run on top of the Aroma VM. Because the Aroma VM is Java-compatible, Gao is not aware of this restriction. Later, when Gao's agent launches a denial-of-service attack which floods the network and begins consuming inordinate amounts of CPU and disk resources, the pattern of misuse is noticed by a Guard, which has been previously authorized to automatically lower the resource limits enforced by the Aroma VM in such situations by one or more policies such as the following:

O+: *IF a Guard notices a pattern of resource misuse by an agent*

THEN that Guard must notify its administrator appropriately

PRECEDENCE A-: The agent is forbidden from using more than 25% of the resource

ELSE A-: The agent is forbidden from using more than 10% of the resource

The policy requires the Guard to notify the administrator who can determine whether this is a false alarm (in which case the agent's resources can be restored by a new policy setting) or whether this is a real attack (in which case the administrator may choose to further lower A's resource usage). If the effort to notify the administrator fails, the Guard is authorized to reduce resource usage to 10% on its own. In this case, transparently reducing resource usage is better than peremptorily terminating the agent because in the former case the agent will be unaware that it's misuse has been detected.

The requirement for the Guard to be able to act autonomously in making an initial response to the attack is akin to the need for a sprinkler system in a building to go off in the presence of smoke before the fire department arrives. Though there is a risk that the signal may have been a false alarm, it is still far better in most cases to have limited the potential damage through prompt action. Moreover, in the case of a malicious agent that is attacking the network, the administrator may not be able to reconfigure a remote sensor until a provisional limit is placed on network resource usage.

A+: *Team A is authorized to use 50% of the CPU*

In order to guarantee a certain quality of service for other agents, Team A is limited to 50% in the amount of CPU resources it is authorized to use. In this example, however, note that the policy says nothing about how the CPU resources should be allocated among members of Team A, so within-team resource allocation is left up to the particular algorithm used by the enforcer performing this task.

3.4.5 Monitoring and response policies

It may sometimes be desirable to represent obligations for the system to perform specific monitoring and response actions as policy:

O+: *IF an authorization failure event occurs*

THEN the authorization mechanism must record the pertinent data in the system log

PRECEDENCE A-: the authorization mechanism is forbidden to perform any other action

ELSE O+: the authorization mechanism must notify the administrator appropriately

In this example, the authorization mechanism is required to record pertinent data in the system log if an authorization failure even occurs. In another example from Ultra*Log:

O+: *IF there is a new defense posture*

THEN the policy applicability condition monitor must deploy the M&R component group for the new defense posture and decactivate the M&R component for previous defense posture

PRECEDENCE A-: the policy applicability condition monitor is forbidden to perform any other action

ELSE O+: the policy applicability condition monitor must notify the administrator appropriately

This policy requires a new set of monitoring and response components to be activated when the defense posture changes (e.g., a change from threatcon alpha to threatcon bravo).

3.4.6 Mobility policies

A-: *Agents that are members of the trust domain are forbidden from moving to host H*

This example illustrates how the movement of software agents from one host to another can be controlled by policy in the same way that any other action is governed providing appropriate enforcement mechanisms are in place.

In a more complex example based on research by Knoll *et al.* ..()()[69], the trust level of a mobile software agent is determined in part by where it has traveled in the past (i.e., there is a greater or lesser possibility that it may have been tampered with by a malicious host). The trust level in turn is used to limit the permissions of the agent in the future:

A-: *Agents are forbidden from performing sensitive action X*
If their trust level <= threshold

The following example, pertinent to our NASA work on the PSA, obligates the PSA to move away from danger:

O+: *IF a situation dangerous to a PSA is present in some location*
THEN the PSA must move out of that location[21]

4 Social Aspects of Agent Acceptability

Building on the basic technical capabilities of KAoS and Nomads services, we are developing policies to facilitate natural and effective human-agent interaction and thus address the social side of agent acceptability.

In this section, we first outline some foundational principles for human-agent interaction (Section 4.1). This is followed by a brief overview of policy-based human-agent teamwork (Section 4.2). Finally, we present examples of policy types relating to social aspects of agent acceptability (Section 4.3).

4.1 Cornerstones of Human-Agent Interaction

Elsewhere we have outlined a preliminary perspective on the basic principles and pitfalls of adjustable autonomy and human-centered teamwork gleaned from the research literature [18]:

- Study teamwork in practice
- Exploit human-agent synergy
- Adjust autonomy

[21] Consistent with Asimov's laws, however, the PSA might be obliged by a higher-level policy to stay if its presence was needed to help a human.

- Continuously expose relevant state
- Avoid clumsy automation

These five cornerstones of human-agent interaction are briefly summarized below.

4.1.1 Study teamwork in practice

Effective study of human-agent interaction necessarily begins with a detailed real-world understanding of how people work in the environment where the agents to be designed will be employed. A traditional task or functional analysis of work leaves out informal logistics, especially how environmental conditions come to be detected and how problems are resolved. Without consideration of these factors, analysts cannot accurately understand how work and information actually flow, nor can they properly design software agents that assist with human tasks. A model of work practice, by way of contrast, focuses on informal, circumstantial, and located behavior in which coordination occurs. In such a model the inevitable discrepancies between plan and reality are highlighted and provide useful insights into "how things really work" [1; 29; 30; 101]. A detailed description of Brahms, which we have used for modeling of teamwork in practice in space settings, may be found in [100].

4.1.2 Exploit human-agent synergy

Within the perspective of human-centered agent teamwork, the differences between people and autonomous agents are best viewed as broad complementarities. The point is not to think so much about which tasks are best performed by humans and which by agents (as in the tradition of Fitts [47] *et al.* and his successors) but rather how tasks can best be shared and performed by both humans and agents working in concert [49; 61].[22]

4.1.3 Adjust autonomy

Approaches to adjustable autonomy are a useful means to enable principled yet dynamic flexibility in the roles of people and agents [42]. To the extent we can adjust agent autonomy with reasonable dynamism (ideally allowing handoffs of control among team members to occur anytime) and with a sufficiently fine-grained range of levels, teamwork mechanisms can flexibly renegotiate roles and tasks among humans and agents as needed when new opportunities arise or when breakdowns occur. Research in adaptive function allocation-the dynamic assignment of tasks among humans and machines-provides some useful lessons for implementations of adjustable autonomy in intelligent systems [60].

4.1.4 Continuously expose relevant state

Maintaining appropriate mutual awareness among team members is paramount for human-agent interaction in complex environments [28; 43]. People need to understand what is happening and why when a teammate tends to respond in a certain way; they need to be able to control the actions of an agent even when it does not always wait for the human's input before

[22]The ultimate in such symbiosis is where the boundary between agents and people disappears altogether, with the agents being subsumed into the human's eudaemonic space (i.e., the agent seems to be part of the person). See Section 4.4 below.

it makes a move; and they need to be able to reliably predict what will happen, even though the agent may alter its responses or adjust its autonomy over time. Humans and agents must be aware of what team members are doing, why they are doing it, and how it relates to the overall accomplishment of the task. When mistakes are made, it must be clear how actors can recover from them [12; 14].

4.1.5 Avoid clumsy automation

In designing human-agent systems, it is important to dispel the misconception that automation is a simple substitution of machine for human effort [28]. Instead it is clear that automated assistance more fundamentally changes the nature of the task itself, sometimes with serious unforeseen consequences. Moreover, although delegation of tasks to an agent might at face value seem to reduce an operator's workload, it usually creates additional burdens such as the need to monitor the performance of the agent. Ignorance of such considerations leads to what Wiener [121] called "clumsy automation."

4.2 Policy-Based Human-Agent Teamwork

Teamwork has become the most widely accepted metaphor for describing the nature of cooperation in multi-agent systems. In most approaches, the key concept is that of shared knowledge, goals, and intentions that function as the glue that binds team members in such systems together [35; 115]. By virtue of a largely reusable explicit formal model of shared intentions, general responsibilities and commitments that team members have to each other are managed in a coherent fashion that facilitates recovery when unanticipated problems arise. For example, a common occurrence in joint action is when one team member fails and can no longer perform in its role. A general teamwork model might entail that each team member be notified under appropriate conditions of the failure, thus reducing the requirement for special-purpose exception handling mechanisms for each possible failure mode.

Whereas early research on agent teamwork focused mainly on agent-agent interaction, teamwork principles are now being formulated in the context of human-agent interaction [15; 18]. Unlike autonomous systems designed primarily to take humans out of the loop, many new efforts are specifically motivated by the need to support close continuous multimodal human-agent interaction [27; 34; 54; 63; 74; 88; 114].

The KAoS policy-based teamwork model defines what constitutes a team and the nature of many of its collaborative activities. The set of policies we are designing for human-robotic interaction go beyond the traditional policy concerns about security and safety in significant ways. As one example, consider how policy can be used to ensure effective communication among team members. Previous research on generic teamwork models has explored this issue to a limited degree within the context of communication required to form, maintain, and abandon joint goals. However, more research is needed to address the complexities of maintaining mutual awareness in human-agent as opposed to agent-agent interaction.

With previous research in agent teamwork, we share the assumption that, to the extent possible, teamwork knowledge should be modeled explicitly and separately from the problem-solving domain knowledge. Policies for agent safety and security as well as context- and culturally-sensitive teamwork behavior can be represented as KAoS policies that enable many aspects of the nature and timing of the agent's interaction with people to be appropriate without requiring each agent to individually encode that knowledge. Agent designers can concen-

trate on developing unique agent capabilities while assuming that many of the basic rules of effective human-agent coordination will be built into the environment as part of the policy infrastructure.

4.3 Examples of Policy Types Relating to Social Aspects of Agent Acceptability

Norman suggests that the social aspects of agent acceptability include such things such providing reassurance that all is working according to plan, providing an understandable and controllable level of feedback about agent's intentions and actions, and accurately conveying the agent's capabilities and limitations [87]. In short, humans must be informed enough to be able to easily step in and help when the situation becomes more than the agent can handle, and agents on their part must be made more competent in conveying the appropriate information to humans and acting in partnership with them. Speaking of the central problems of conventional automation, Norman writes:

> "The problem. . . is that automation is at an intermediate level of intelligence, powerful enough to take over control that used to be done by people, but not powerful enough to handle all abnormalities. Moreover, its level is insufficient to provide the continual, appropriate feedback that occurs naturally among human operators. To solve this problem, the automation should either be made less intelligent or more so, but the current level is quite inappropriate... . . Problems result from inappropriate application, not overautomation" [85].

In contrast to the examples of technical policies in Section 3.4 above, our work to begin encoding these social issues in policy is relatively recent and is likely to evolve considerably in the near future. Some of this will require the resolution of difficult research issues; we are beginning implementation with those policies that are most straightforward and will then continue to progress incrementally to more complex ones. We will give examples from six categories of policy: organization (4.3.1), notification (4.3.2), conversation (4.3.3), nonverbal expression (4.3.4), collaboration (4.3.5), and adjustable autonomy (4.3.6).

4.3.1 Organization policies

Some policy management systems, in part as an artifact of their mode of policy representation, require many or all of what we call organization policies to be represented as "meta level," "higher order," or other sorts of special policies. In KAoS, many of these can be specified uniformly in the same way that other kinds of policies are represented.[23]

A+: *Individuals of the class Domain Manager are permitted to approve policies*

The KAoS actor ontology distinguishes between people and various classes of agents. Most agents can only perform *ordinary actions*, however various components that are part of the infrastructure (e.g., domain manager, guard) as well as authorized human users may variously be permitted or obligated to perform *policy actions*, such as policy approval and enforcement.

[23] An exception is delegation policies, which are not yet supported in KAoS.

A+: *Any person in the Manager Role is permitted to authorize check payment IF the same person is not also the check issuer*

The example specifies a dynamic separation of duty, where the issuer of the check is not allowed to also be the one who authorizes payment on that check.

A-: *An agent is forbidden to register to domain D IF it is already registered to any individual of class domain*

A-: *An agent is forbidden to register to any individuals of class domain IF it is already registered to domain D*

The pair of policies above specify that an agent cannot simultaneously be registered both as a member of domain D and also of some other domain.

4.3.2 Notification Policies

Building on the work of [96; 97], we are developing KAoS notification policies in the context of our NASA applications. The vision of future human-agent interaction is that of loosely coordinated groups of humans and agents. As capabilities and opportunities for autonomous operation grow in the future, agents will perform their tasks for increasingly long periods of time with only intermittent supervision. Most of the time routine operation is managed by the agents while the human crews perform other tasks. Occasionally however, when unexpected problems or novel opportunities arise, people must assist the agents. Because of the loose nature of these groups, such communication and collaboration must proceed asynchronously and in a mixed-initiative manner. Humans must quickly come up to speed on situations with which they may have had little involvement for hours or days. Then they must cooperative effectively and naturally with the agents as true team members.[24] Hence the challenge of managing notification and situation awareness for the crewmembers.

Various ontologies supporting notification (e.g., basic concepts for categories of events, roles, notifications, latency, focus of attention, and presence) form the foundation of this work. In conjunction with these ontologies, notification policies and their parameter settings are created, as in the example below:

O+: *IF new notification = true AND utility >= notifyThreshold AND utility < doItThreshold THEN notify the space station crewmember appropriately*

Human attention is a scarce resource. When an important event is signaled, the utility of various alternatives (e.g., notify the crewmember, perform some required action without interrupting the person, or do nothing) is evaluated. If a notification is required and the current task is well-defined, the KAoS-Brahms infrastructure will take into account the task and other contextual factors to perform the notification in a manner that is context-appropriate with respect to modality, urgency, and location of the human. Because such knowledge resides in the infrastructure rather than as part of the knowledge of each agent, agent development is simplified.

[24] Actually, this vision points to two major opportunities for policy-based help: 1. the use of policy to assure that unsupervised autonomous agent behavior is kept within safe and secure limits of prescribed bounds, even in the face of buggy or malicious agent code; and 2. the use of policy to assure effective and natural human-agent team interaction, without individual agents having to be specifically programmed with the knowledge to do so.

4.3.3 Conversation Policies

Explicit conversation policies simplify the work of both the agent and the agent designer [16; 17; 57; 58]. In comparison to unrestricted agent dialogue, conversation policies reduce the agents' inferential burden by limiting the space of alternative conversational productions and parameters that they need to consider both in generating messages and in interpreting messages received from other agents. Because a significant measure of conversational planning for routine interactions can be encoded in conversation policies offline and in advance, the agents can devote more of their computational power at runtime to other things.

O+: *IF response lag of conversation X > M minutes*

THEN the agent must terminate conversation X

This conversation policy requires the agent to unilaterally terminate a conversation when a lag of M minutes has elapsed in waiting for a response, preventing conversations from staying open indefinitely.

A-: *Agents are forbidden to send a message of any type other than Reply*

IF conversation type of the conversation is Request-Reply AND the previous message type of the conversation is Request

This conversation policy enforces a sequence of messages in the Request-Reply conversation type such that a message of type *request* must always be followed by a message of type *reply*.

A+: *Agents are permitted to send TRANSCOM messages with return receipts*

This conversation policy example from the Ultra*Log application, allows the sending of TRANSCOM messages that require return receipts. Note that this policy would only be appropriate in a tyrannical domain that prohibited all messages that were not explicitly permitted.

More complex sorts of policies dealing, for example, with Clark's concept of common ground [31] or improvisational approaches to conversation [93], will also be important in effective human-agent interaction. Though such policies go beyond what is possible in the current version of KAoS, we expect to begin addressing them as part of a collaboration with Allen *et al.* [2; 3] in the near future.

4.3.4 Nonverbal Expression Policies

Where possible, agents usually take advantage of explicit verbal channels for communication in order to reduce the need for relying on current primitive robotic vision and auditory sensing capabilities [84, p. 295]. On the other hand, animals and humans often rely on visual and auditory signals in place of explicit verbal communication for many aspects of coordinated activity. As part of our work on human-robotic interaction for NASA, the Army, and the Navy, we are developing policies to govern various nonverbal forms of expression in hardware and software agents. These nonverbal behaviors will be designed to express not only the current state of the agent but, importantly, also to provide rough clues about what it is going to do next. In this way, people can be better enabled to participate with the agent in

coordination, support, avoidance, and so forth. In this sense, nonverbal expressions are an important ingredient in enabling human-agent teamwork.

Maes and her colleagues were among the first to explore this possibility in her research on software agents that continuously communicated their internal state to the user via facial expressions (e.g., thinking, working, suggestion, unsure, pleased, and confused) [80]. Breazeal has taken inspiration from the research in child psychology [116] to develop robot displays that reflected four basic classes of preverbal social responses: affective (changing facial expressions), exploratory (visual search, maintenance of mutual regard with human), protective (turning head away), and regulatory (expressive feedback to gain caregiver attention, cyclic waxing and waning of internal states, habituation, and signals of internal motivation) [22]. Books on human etiquette [118] contain many descriptions of appropriate behavior in a wide variety of social settings. Finally, in addition to this previous work, we think that display behavior among people [83] and groups of animals will be one of the most fruitful sources of policy for effective nonverbal expression in agents. Our initial study indicates that there are useful agent equivalents for each of Smith's ten categories of widespread vertebrate animal cooperation and coordination displays [102, pp. 84-125].

O+: *IF the current task of the PSA is of type uninterruptible*

THEN the PSA must blink red light until the current task is finished

PRECEDENCE: A-: The PSA is forbidden from performing any tasks but the current one

This policy requires the PSA to blink a red light while it is busy performing an uninterruptible task. During this time, it is also forbidden from performing any tasks but the current one. Related messages it may want to give with a similar signal might include: "I am unable to make contact with anybody," "Do not attempt to communicate with me (for whatever reason, e.g., "my line is bugged")." On the positive side, various uses of a green light might signal messages such as: "I am open for calls," "I need to talk to someone," or "May I interject something into this conversation? " Displays in this general interactional category clearly have benefits for coordination among groups of agents by providing information about which are or are not in a position to interact with others, in what ways, when, and so forth.

O+: *IF a conversation has been initiated with someone*

THEN the PSA must face the one with whom it is conversing, so long as they are within the line of sight, until the conversation has finished

This policy implements a kind of display associated with maintaining a previously established association. This display might be especially useful when the PSA is moving around the room and needs to let a person know that it is still attending to the ongoing conversation.

O+: *IF the current task of the PSA is to move some distance greater than D*

THEN the PSA must signal its intention to move for S seconds

PRECEDENCE: A-: The PSA is forbidden from executing its move

It's no fun being hit in the head by a flying robot that suddenly decides to go on the move. This policy prevents the PSA from moving until it has first signaled its intention to move for some number of seconds. Besides the pre-move signaling, some kind of signaling could also take place during the move itself. In addition to this movement signaling policy, other policies should be put in place to require the PSA to stay at a safe and comfortable distance from people, other robotic agents, and space station structures and equipment.

4.3.5 Collaboration Policies

O+: *IF an agent becomes aware that a team goal has been achieved, or has become unachievable or irrelevant*

THEN the agent must notify the other team members in an appropriate manner

PRECEDENCE: A-: The team member is forbidden from actions that are performed only in order to achieve the former team goal

A similar version of this policy is one of the centerpieces of the classic theory of teamwork originally proposed by Cohen and Levesque [35]. Though there is potentially a lot of complexity buried in the machinery that determines whether the conditional is true, the policy imperative that results from this conditional is relatively simple by way of contrast and can be represented straightforwardly in KAoS. All the foundational ontologies and mechanisms developed to support other kinds of notification policies can also be brought to bear in this context. In this sense, the example can be seen as just a special kind of notification policy.

O+: *IF an agent suspends work on a current task in order to attend to a new higher priority task*

THEN the agent must notify the other actors involved in an appropriate manner

PRECEDENCE: A-: The agent is forbidden from executing its new task

Just as a sudden physical move might surprise other actors unless it is appropriately signaled in advance, so an unexpected change in current task might be jarring to others unless it is heralded in some fashion. Note that this policy presupposes that additional actors beyond the team members themselves that share a joint goal may require notification.

4.3.6 Adjustable Autonomy Policies

Humans and agents may play mutual roles that vary according to the relative degree of initiative appropriate to a given situation (Figure 12).[25] At the one extreme, traditional systems are designed to carry out the explicit commands of humans with no ability to ignore orders (i.e., executive autonomy [8; 45]), generate their own goals (i.e., goal autonomy [45; 78]), or otherwise act independently of environmental stimuli (i.e., environmental autonomy [21; 45]). Such systems cannot, in any significant sense, act; they can only be acted upon. At the other end of the spectrum is an imagined extreme in which agents would control the actions of humans.[26] Between these two extremes is the domain of today's agent systems, with most agents typically playing fixed roles as servants, assistants, associates, or guides. Such autonomous

[25] For a more fine-grained presentation of a continuum of control between humans and machines, see Hancock and Scallen's [60] summary of Sheridan's ten-level formulation. Robert Taylor (personal communication) surmises from his experience that this may be far more levels that what are useful in practice. Barber *et al.* differentiate three kinds of relationships among agents: command-driven (i.e., the agent is fully subordinated to some other agent), true consensus (i.e., decision-making control is shared equally with other agents), and locally autonomous/master (i.e., the agent makes decisions without consulting other agents and may be allowed to command subordinates) [7].

[26] Of course, in real systems, the relative degree of initiative that could be reasonably taken by an agent or human would not be a global property, but rather relative to particular functions that one or the other was currently assuming in some context of joint work (see [8; 13; 56; 60]).

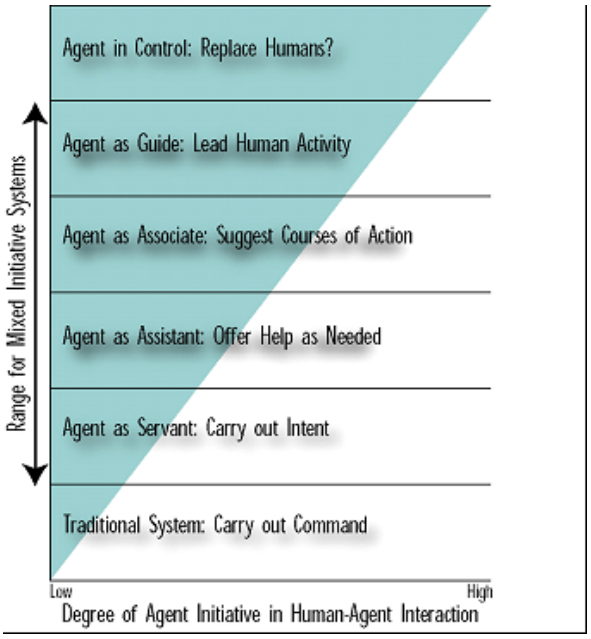

Figure 12: Spectrum of agent roles in human-agent interaction [15].

systems are designed with fixed assumptions about what degree of initiative is appropriate to their tasks. They execute their instructions without considering that the optimal level of autonomy may vary by task and over time, or that unforeseen events may prompt a need for either the human or the agent to take more control. At the limit of this extreme are strong, silent systems [92] with only two modes: fully automatic and fully manual. In practice this can lead to situations of human "underload," with the human having very little to do when things are going along as planned, followed by situations of human "overload," when extreme demands may be placed on the human in the case of agent failure.

Although in practice many do not live up to their billing, the design goal of mixed-initiative systems is to allow agents to dynamically and flexibly assume a range of roles depending on the task to be performed and the current situation [24; 36; 46]. Research in adjustable autonomy supports this goal through the development of an understanding of how to ensure that, in a given context, the agents are operating at an optimal boundary between the initiative of the human and that of the agents. People want to maintain that boundary at a sweet spot in the tradeoff curve that minimizes their need to attend to interaction with the agent while providing them with a sufficiently comfortable level of assurance that nothing will go wrong.

In principle, the actual adjustment of an agent's level of autonomy could be initiated either by a human, the agent itself, or some trusted software component.[27] There are several dimen-

[27] Cohen [37] draws a line between those approaches in which the agent itself wholly determines the mode of interaction with humans (mixed-initiative) and those where this determination is imposed externally (adjustable autonomy). Additionally, mixed-initiative systems are considered by Cohen to generally consist of a single user and a single agent. However, it is clear that these two approaches are not mutually exclusive and that, in an ideal world, agents would be capable of both reasoning about when and how to initiate interaction with the human and also of subjecting themselves to the external direction of whatever set of explicit authorization and obligation policies were currently in force to govern that interaction. Additionally, there is no reason to limit the notion of "mixed initiative" systems to the single agent-single human case. Hence we prefer to think of mixed-initiative systems as being those systems that are capable of making context-appropriate adjustments to their level of

sions of this level that can be varied such as: 1) type or complexity of tasks or functions it is permitted to execute, 2) which of its functions or tasks may be autonomously controlled, 3) circumstances under which the agent will override manual control, 4) duration of autonomous operation, and 5) the circumstances under which a human may be interrupted (or must be interrupted) in order to provide guidance [42].

When evaluating options for adaptively reallocating tasks among team members, it must be remembered that dynamic role adjustment comes at a cost. Measures of expected utility can be used to evaluate the tradeoffs involved in potentially interrupting the ongoing activities of agents and humans in such situations to communicate, coordinate, and reallocate responsibilities [37; 63; 64]. It is also important to note that the need for adjustments may cascade in complex fashion: interaction may be spread across many potentially-distributed agents and humans who act in multiply-connected interaction loops. For this reason, adjustable autonomy may involve not merely a simple shift in roles among a human-agent pair, but rather the distribution of dynamic demands across many coordinated actors.[28] Defining explicit policies for the transfer of control among team members and for the resultant required modifications to coordination constraints can prove useful in managing such complexity [94].

O+: *IF elapsed time since last report > time T*

THEN the agent must notify the human supervisor about its status in the appropriate manner

This simple policy sets the duration of autonomous operation for an agent, requiring it to notify a human supervisor about its status at predetermined intervals.

A+: *Space station crewmembers in the Trusted Operator Role are permitted to override PSA non-critical negative authorizations*

Sometimes it is critical for authorized human operators to be able to immediately countermand some negative authorization of an agent (i.e., allowing it to do things which it normally is not authorized to do). While this could be done instead by modifying the policy in the usual way, sometimes it is more practical to do this directly by overriding a prohibition on a one-time temporary basis. However, overriding certain operations (e.g., flying the PSA into a wall of the space station) may require consent of both the space station commander and an authorized person at mission control.

O+: *IF no crewmember is monitoring the environment in space station module X*

THEN PSA must monitor environment in module X

PRECEDENCE: A+: PSA is permitted to monitor the environment in module X

social autonomy (i.e., their level or mode of engagement with the human), whether a given adjustment is made as a result of reasoning internal to the agent or due to externally-imposed policy-based constraints.

[28] As Hancock and Scallen [60] rightfully observe, the problem of adaptive function allocation is not merely one of efficiency or technical elegance. Economic factors (e.g., can the task be more inexpensively performed by humans, agents, or some combination?), political and cultural factors (e.g., is it acceptable for agents to perform tasks traditionally assigned to humans?), or personal and moral considerations (e.g., is a given task enjoyable and challenging vs. boring and mind-numbing for the human?) are also essential considerations.

Sometimes the PSA may be required to temporarily take upon itself functions that human crewmembers would normally provide. Here the PSA is both given permission and an obligation to monitor the environment in module X if a crewmember is not currently doing so. Similar policies could come into play when a crewmember becomes overloaded or injured such that the crewmember no longer is able to perform the task within predetermined criteria. In such cases, agents could be authorized and/or obligated to assist.

4.4 Cognitive and Robotic Prostheses

For some researchers, the ultimate in human-agent teamwork is the notion of agents that can function as extensions of the human brain *(cognitive prostheses)* and body *(robotic prostheses)* [49; 59; 62]. In this section we briefly suggest some preliminary considerations relating to human-agent interaction in the development of such capabilities.

At the outset, we recognize humans are an advantaged lot, each of us having been endowed with a "good brain and an unspecialized body" [83, p. 489], which means that we are in a better position than any other creature to make and use a variety of tools. Moreover, bipedal locomotion has always had the beneficial side effect of freeing one hand to explore the environment and the other to wield those tools. Ford *et al.* argue that the accumulated tools of human history can all profitably be regarded as prostheses, not in the sense that they compensate for the specific disabilities of any given individual [38], but rather because they enable us to overcome the biological limitations shared by all of us: with reading and writing anyone can transcend the finite capacity of human memory; "with a power screwdriver anyone can drive the hardest screw; with a calculator, anyone can get the numbers right; with an aircraft anyone can fly to Paris; and with Deep Blue, anyone can beat the world chess champion" [49].

The prosthetic perspective can be contrasted with the traditional focus of Artificial Intelligence on standalone machine competence and its resulting preoccupation with the Turing Test as its measure of success [51]. Instead, argues Ford, we should start from a human-centered perspective. This implies that we must shift our goal "from making artificial superhumans who can replace us to making superhumanly intelligent artifacts that can amplify and support our own cognitive abilities" [59, p. 61]. We don't need to jettison the acronym of AI, so long as we now take it to refer to the human's *Augmented* Intelligence.[29]

Eyeglasses, a well-known example of an ocular prosthesis,[30] provide a particularly useful example of three foundational concepts that are important to an understanding of cognitive and robotic prostheses:

1. *Transparency.* "Eyeglasses leverage and extend our ability to see, but in no way model our eyes: They don't look or act like them and wouldn't pass a Turing test for being an eye" [59, p. 61]. A key feature of eyeglasses is that they can be used more or less

[29]The 1962 report of Engelbart entitled *Augmenting Human Intellect* presciently stressed the theme of "improving the intellectual effectiveness of the individual human being...through extensions of means developed and used in the past to help man apply his native sensory, mental, and motor capabilities [Like] most systems its performance can best be improved by considering the whole as a set of interacting components rather than by considering the components in isolation" [44, pp. 1-2].

[30]The notion of augmenting sight through eyeglasses was "first mentioned by Roger Bacon in 1268. In the 1665 preface to Micrographia, Robert Hooke goes further, suggesting the addition 'of artificial Organs to the natural...to improve our other senses of hearing, smelling, tasting, and touching" [103].

transparently—even forgetting they are present—just as humans with myopia don't think constantly about the wearing of the contact lenses but rather about the fact that they are seeing more effectively *through* them.[31]

2. *Unity.* Since our goal is not making smart eyeglasses, but rather augmenting the human's ability to see, the minimum unit of discussion for the design of a prosthesis includes the device, the human, and the environment in which the human will use the device. This mode of analysis necessarily blurs the line between humans and technology.[32]

3. *Fit.* Your eyeglasses won't fit me; neither will mine do you any good. Prostheses must fit the human and technological components together in ways that synergistically exploits their mutual strengths and mitigates their respective limitations. This implies the requirement for a rich knowledge of how humans function.[33]

The elaboration of foundational concepts that are important to an understanding of cognitive and robotic prostheses and the study of human functions in particular environments happily dovetail with progress in the miniaturization of computing devices and the formulation of design principles for wearable computing [89]. Mann [81] was among the first to elucidate some of the necessary criteria for devices to be successfully subsumed into the human's eudaemonic space (i.e., where the device seems to be part of the person).[34] Mann describes three required operational modes for wearable computing:

- **Constancy:** The computer runs continuously, and is *always ready* to interact with the user. Unlike a handheld device, laptop computer, or PDA, it does not need to be opened up and turned on prior to use. The signal flow from human to computer, and computer to human, . . . , runs continuously to provide a constant user interface.

[31]The manner in which perception operates during the use of good tools was insightfully described many years ago by Polanyi: "When we use a hammer to drive a nail, we attend to both nail and hammer, but in a different way. We watch the effect of our strokes on the nail and try to wield the hammer so as to hit the nail most effectively. When we bring down the hammer we do not feel that its handle has struck our palm but that its head has struck the nail. Yet in a sense we are certainly alert to the feelings in our palm and the fingers that hold the hammer. They guide us in handling it effectively, and the degree of attention that was given to the nail is given to the same extent but in a different way to these feelings. The difference may be stated by saying that the latter are not, like the nail, objects of our attention, but instruments of it. They are not watched in themselves; we watch something else while keeping intensely aware of them. I have a subsidiary awareness of the feeling in the palm of my hand which is merged into my focal awareness of my driving in the nail"[90, p. 55].

[32]In 1960, Licklider [73] introduced the concept of *man-computer symbiosis:* "the hope is that, in not too many years, human brains and computing machines will be coupled together very tightly and that the resulting partnership will think as no human brain has ever thought and process data in a way not approached by the information-handling machines we know today."

[33]A good example of this is the OZ cockpit display [59; 104]. Through a groundbreaking study on the limits of human central and peripheral vision, IHMC's David Still discovered that peripheral vision can pick up 10 times the amount of detail than previously thought. Using this finding, he tailored the design of stimuli in a cockpit display to exploit the human sensory system's natural filtering and processing capabilities and manipulate the data so it provides exactly what the pilot needs to know at any particular time. Stunningly, the OZ cockpit display is completely void of dials and gauges of ordinary cockpits, yet is easier to learn, more straightforward to control, and more robust to temporary visual system impairment.

[34]Mann's formulations have evolved over time. Here we discuss the version that can be found at http://www.eyetap.org/defs/glossary/wearcomp. See also Thad Starner's thorough survey of the field in his dissertation on Wearable Computing [103].

- **Augmentation:** Traditional computing paradigms are based on the notion that computing is the primary task. Wearable computing, however, is based on the notion that computing is *not* the primary task. The assumption of wearable computing is that the user will be doing something else at the same time as doing the computing. Thus the computer should serve to augment the intellect or augment the senses, ...

- **Mediation:** Unlike [traditional computers], the wearable computer can encapsulate us. It doesn't necessarily need to completely enclose us. There are two aspects to this encapsulation:

  i. **Solitude:** It can function as an information filter, and allow us to block out material we might not wish to experience, ..., [or] it may simply allow us to alter our perception of reality.

  ii. **Privacy:** Mediation allows us to block or modify information leaving our encapsulated space. In the same way that ordinary clothing prevents others from seeing our naked bodies, the wearable computer may, for example, serve as an intermediary for interacting with untrusted systems.

Because of its ability to encapsulate us, ..., [wearable computing devices] may also be able to make measurements of various physiological quantities.

Besides these operational modes, Mann describes six attributes of wearable systems:

- **Unmonopolizing** of the user's attention, ..., [though it may] mediate (augment, alter, or deliberately diminish) the sensory capabilities.

- **Unrestrictive** to the user: ..., 'you can do other things while using it' ....

- **Observable** by the user: it can get your attention continuously if you want it to ....

- **Controllable** by the user ....

- **Attentive** to the environment: it is environmentally aware, multimodal, multi-sensory ..., [thus increasing] the user's [situation] awareness ....

- **Communicative** to others: it can be used as a communications medium ....

DARPA's Augmented Cognition (AugCog) Program (http://www.darpa.mil/ito/research/ac/) is an example of an early effort focused on appropriately exploiting and integrating all available channels of communication from agents to the human (e.g., visual, auditory, tactile) and conversely sensing and interpreting a wide range of physiological measures of the human in real-time so they can be used to tune agent behavior and thus enhance joint human-machine performance.[35] For example, in IHMC's Adaptive Multi-Sensory Integration (AMI) augmented cognition prototype sets of system sensor agents (e.g., joystick), human sensor

[35] A related program focused on similar issues with a robotics emphasis is NSF's Robotics and Human Augmentation (http://www.interact.nsf.gov/cise/descriptions.nsf/5b8c6c912ebf7f9b8525662c00723201/5e8661fa698fe674852565d9005985ef?OpenDocument). See also DARPA's Mobile Autonomous Robot Software (MARS) Robotic Vision 2020 Program (http://www.darpa.mil/ito/solicitations/FBO_02-15.html).

agents (e.g., EEG, pupil tracking, arousal meter), human display agents (e.g., visual, auditory, tactile), and adaptive automation agents (e.g., performing specific flight tasks) could work together with a pilot to promote stable and safe flight, sharing and adjusting aspects of control among the human and virtual crew member agents while taking system failures and human attention and stress loads into account.

While it is still too early to gauge the success of efforts such as AugCog, let alone to prescribe detailed principles for making cognitive and robotic prostheses acceptable to humans, it is clear that such modes of interaction will require new ways of thinking about human-agent interaction. In an insightful essay called *The Teddy* [86], Norman discusses some of the issues and implications of the widespread long-term habitual use of such technologies:

- Would we get so dependent that we would become disoriented without them?
- If they are constantly recording every event, should we allow them to be turned off? To protect civil liberties, you must be able to, and an indicator must show if someone's device is listing to you.
- Should it be programmed to always be supporting and encouraging (thus removing us from reality) or criticism and correction (thus reminding us of a nagging parent)? Getting the right balance is difficult in human relationships, how can we expect technology designers to do better?
- If we are never alone, when would we think? Would this accelerate the already tuned-out tendencies of headphone wearers?

5 Conclusions

In this chapter, we have outlined some of the technical and social challenges in the problem of making agents acceptable to people and given examples and explanation of how a policy-based approach might be used to address some of those challenges. We hope that these initial efforts will inspire others to devote greater attention to reusable models and tools to assure the security, safety, naturalness, and effectiveness of human-agent teams.

Acknowledgements

The authors gratefully acknowledge the support of the DARPA CoABS, Augmented Cognition, DAML, and Ultra*Log Programs, the NASA Cross-Enterprise and Intelligent Systems Programs, the Army Research Lab, the Office of Naval Research, and the National Technology Alliance, and Fujitsu Labs while preparing this paper. We are also grateful for the contributions of James Allen, Alessandro Acquisti, Guy Boy, Kathleen Bradshaw, Mark Burstein, Murray Burke, Alberto Canas, Bill Clancey, Rob Cranfill, Rich Feiertag, Ken Ford, Yuri Gawdiak, Mark Greaves, Jack Hansen, Pat Hayes, Wayne Jansen, Mike Kerstetter, Mike Kirton, Shri Kulkarni, Henry Lieberman, James Lott, Frank McCabe, Cindy Martin, Robin Murphy, Nicola Muscettola, Jerry Pratt, Debbie Prescott, Timothy Redmond, Sue Rho, Sebastien Rosset, Dylan Schmorrow, Debbie Schrekenghost, Kent Seamons, Mike Shafto, Maarten Sierhuis, Milind Tambe, Austin Tate, Ron Van Hoof, and Tim Wright.

References

[1] A. Acquisti, M. Sierhuis, W. J. Clancey, & J. M. Bradshaw, (2002). Agent-based modeling of collaboration and work practices onboard the International Space Station. *Proceedings of the Eleventh Conference on Computer-Generated Forces and Behavior Representation*. Orlando, FL,

[2] J. Allen, D. K. Byron, M. Dzikovska, G. Ferguson, L. Galescu, & A. Stent, (2000). An architecture for a generic dialogue shell. *Journal of Natural Language Engineering*, 6(3), 1-16.

[3] J. F. Allen, D. K. Byron, M. Dzikovska, G. Ferguson, L. Galescu, & A. Stent, (2001). Towards conversational human-computer interaction. *AI Magazine*, 22(4), 27-35.

[4] D. Allsopp, P. Beautement, J. M. Bradshaw, E. Durfee, M. Kirton, C. Knoblock, N. Suri, A. Tate, & C. Thompson, (2002). Coalition Agents eXperiement (CoAX): Multi-agent cooperation in an international coalition setting. *A. Tate, J. Bradshaw, and M. Pechoucek (Eds.), Special issue of IEEE Intelligent Systems*, 17(3), 26-35.

[5] R.Ambrose, C. Culbert, & F. Rehnmark, (2001). An experimental investigation of dexterous robots using Eva tools and interfaces. *AIAA, 4593*.

[6] I. Asimov, (1942/1968). Runaround. In I. Asimov (Ed.), *I, Robot.* (pp. 33-51). London, England: Grafton Books. Originally published in *Astounding Science Fiction*, 1942, pp. 94-103.

[7] K. S. Barber, M. Gamba, & C. E. Martin, (2002). Representing and analyzing adaptive decision-making frameworks. In H. Hexmoor, C. Castelfranchi, & R. Falcone (Ed.), *Agent Autonomy.* (pp. 23-42). Dordrecht, The Netherlands: Kluwer.

[8] K. S. Barber, & C. E. Martin, (1999). Agent autonomy: Specification, measurement, and dynamic adjustment. *Proceedings of the Workshop on Autonomy Control Software, International Conference on Autonomous Agents*. Seattle, WA.

[9] T. Berners-Lee, J. Hendler, & O. Lassila, (2001). The semantic web. *Scientific American.*

[10] C. E. Billings, (1997). Issues concerning human-centered intelligence systems: What's 'human-centered' and what's the problem? *National Science Foundation Workshop on Human-Centered Systems: Information, Interactivity, and Intelligence*. Arlington, VA, http://www.ifp/uiuc.edu/nsfhcs/abstracts/,

[11] M. Boman, (1999). Norms in artificial decision-making. *Artificial Intelligence and Law*, 7, 17-35.

[12] G. Boy, (1998). *Cognitive Function Analysis*. Stamford, CT: Ablex Publishing.

[13] G. Boy, (2002). Human-centered design of artificial agents: A cognitive function analysis approach. In J. M. Bradshaw (Ed.), *Handbook of Software Agents.* (pp. in press). Cambridge, MA: AAAI Press/The MIT Press.

[14] G. Boy, (2002). Interfaces procdurales. *Proceedings of Interaction Homme-Machine (IHM '02)*. Poitiers, France, New York, NY: ACM Press.

[15] J. M. Bradshaw, G. Boy, E. Durfee, M. Gruninger, H. Hexmoor, N. Suri, M. Tambe, M. Uschold, & J. Vitek, (Ed.). (2002). *Software Agents for the Warfighter. ITAC Consortium Report.* Cambridge, MA: AAAI Press/The MIT Press.

[16] J. M. Bradshaw, S. Dutfield, P. Benoit, & J. D. Woolley, (1997). KAoS: Toward an industrial-strength generic agent architecture. In J. M. Bradshaw (Ed.), *Software Agents.* (pp. 375-418). Cambridge, MA: AAAI Press/The MIT Press.

[17] J. M. Bradshaw, M. Greaves, H. Holmback, W. Jansen, T. Karygiannis, B. Silverman, N. Suri, & A. Wong, (1999). Agents for the masses: Is it possible to make development of sophisticated agents simple enough to be practical? *IEEE Intelligent Systems*(March-April), 53-63.

[18] J. M. Bradshaw, M. Sierhuis, A. Acquisti, P. Feltovich, R. Hoffman, R. Jeffers, D. Prescott, N. Suri, A. Uszok, & R. Van Hoof, (2002). Adjustable autonomy and human-agent teamwork in practice: An interim report on space applications. In H. Hexmoor, R. Falcone, & C. Castelfranchi (Ed.), *Agent Autonomy.* (pp. in press). Kluwer.

[19] J. M. Bradshaw, N. Suri, M. R. Breedy, A. Canas, R. Davis, K. M. Ford, R. Hoffman, R. Jeffers, S. Kulkarni, J. Lott, T. Reichherzer, & A. Uszok, (2002). Terraforming cyberspace. In D. C. Marinescu & C. Lee (Ed.), *Process Coordination and Ubiquitous Computing.* (pp. 165-185). Boca Raton, FL: CRC Press. Updated and expanded version of an article that originally appeared in IEEE Intelligent Systems, July 2001, pp. 49-56.

[20] J. M. Bradshaw, A. Uszok, R. Jeffers, N. Suri, P. Hayes, M. H. Burstein, A. Acquisti, B. Benyo, M. R. Breedy, M. Carvalho, D.Diller, M. Johnson, S. Kulkarni, J. Lott, M. Sierhuis, & R. Van Hoof, (2003). Representation and reasoning for DAML-based policy and domain services in KAoS and Nomads. *Proceedings of the Autonomous Agents and Multi-Agent Systems Conference (AAMAS 2003)*. Melbourne, Australia, New York, NY: ACM Press,

[21] S.Brainov, & H. Hexmoor, (2002). Quantifying autonomy. In H. Hexmoor, C. Castelfranchi, & R. Falcone (Ed.), *Agent Autonomy.* (pp. 43-56). Dordrecht, The Netherlands: Kluwer.

[22] C. Breazeal, & B. Scassellati, (1999). How to build robots that make friends and influence people. *IROS.* Kyonjiu, Korea,

[23] R. R. Burridge, J. Graham, K. Shillcutt, R. Hirsh, & D. Kortenkamp, (2003). Experiments with an EVA assistant robot. *Proceedings of the Seventh International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*. Nara, Japan,

[24] M. H. Burstein, & D. V. McDermott, (1996). Issues in the development of human-computer mixed-initiative planning. In B. Gorayska & J. L. Mey (Ed.), *Cognitive Technology: In Search of a Humane Interface.* Elsevier Science.

[25] C. Canto, & O. Faliu, (n.d.). *The History of the Future: Images of the 21st Century*. Paris: Flammarion.

[26] K. Capek, (1920/1990). R. U. R. (Rossum's Universal Robots). In P. Kussi (Ed.), *Toward the Radical Center: A Karel Capek Reader.* North Haven, CT: Catbird Press.

[27] H. Chalupsky, Y. Gil, C. A. Knoblock, K. Lerman, J. Oh, D. V. Pynadath, T. A. Russ, & M. Tambe, (2002). Electric Elves: Agent technology for supporting human organizations. *AI Magazine*, 2, Summer, 11-24.

[28] K. Christofferson, & D. D. Woods, (2002). How to make automated systems team players. In E. Salas (Ed.), *Advances in Human Performance and Cognitive Engineering Research, Vol. 2*. JAI Press, Elsevier.

[29] W. J. Clancey, (2002). Simulating activities: Relating motives, deliberation, and attentive coordination. *Cognitive Systems Review, special issue on Situated and Embodied Cognition.*

[30] W. J. Clancey, (2003). Roles for agent assistants in field science: Understanding personal projects and collaboration. *IEEE Transactions on Systems, Man, and Cybernetics*.

[31] H. H. Clark, (1992). *Arenas of Language Use*. Chicago, IL: University of Chicago Press.

[32] R. Clarke, (1993-1994). Asimov's laws of robotics: Implications for information technology, Parts 1 and 2. *IEEE Computer*, December/January, 53-61/57-66.

[33] J. Clute, & P. Nicholls, (1995). Grolier Science Fiction: The Multimedia Encyclopedia of Science Fiction (CD-ROM). In Danbury, CT: Grolier Electronic Publishing.

[34] P. Cohen, R. Coulston, & K. Krout, (2002). Multimodal interaction during multiparty dialogues: Initial results. *Proceedings of the Fourth IEEE International Conference on Multimodal Interfaces*. Pittsburgh, PA,

[35] P. R. Cohen, & H. J. Levesque, (1991). *Teamwork*. Technote 504. Menlo Park, CA: SRI International, March.

[36] R. Cohen, C. Allaby, C. Cumbaa, M. Fitzgerald, K. Ho, B. Hui, C. Latulipe, F. Lu, N. Moussa, D. Pooley, A. Qian, & S. Siddiqi, (1998). What is initiative? *User Modeling and User-Adapted Interaction*, 8(3-4), 171-214.

[37] R. Cohen, & M. Fleming, (2002). Adjusting the autonomy in mixed-initiative systems by reasoning about interaction. In H. Hexmoor, C. Castelfranchi, & R. Falcone (Ed.), *Agent Autonomy.* (pp. 105-122). Dordrecht, The Netherlands: Kluwer.

[38] E. Cole, & P. Dehdashti, (1998). Computer-based cognitive prosthetics: Assistive technology for the treatment of cognitive disabilities. *Proceedings of the Third International ACM Conference on Assistive Technologies (ACM SIGCAPH - Computers and the Physically Handicapped)*. Marina del Rey, CA,

[39] R. Conte, & C. Castelfranchi, (1995). *Cognitive and social action.* London, England: UCL Press.

[40] M. d'Inverno, & M. Luck, (2001). *Understanding Agent Systems*. Berlin, Germany: Springer-Verlag.

[41] N. Damianou, N. Dulay, E. C. Lupu, & M. S. Sloman, (2000). *Ponder: A Language for Specifying Security and Management Policies for Distributed Systems, Version 2.3*. Imperial College of Science, Technology and Medicine, Department of Computing, 20 October 2000.

[42] G. Dorais, R. P. Bonasso, D. Kortenkamp, B. Pell, & D. Schrekenghost, (1999). Adjustable autonomy for human-centered autonomous systems on Mars. *Proceedings of the AAAI Spring Symposium on Agents with Adjustable Autonomy. AAAI Technical Report SS-99-06*. Menlo Park, CA, Menlo Park, CA: AAAI Press,

[43] M. R. Endsley, & D. J. Garland, (Ed.). (2000). *Situation Awareness Analysis and Measurement.* Englewood Cliffs, NJ: Lawrence Erlbaum.

[44] D. C. Engelbart, (1962). *Augmenting Human Intellect: A Conceptual Framework*. Air Force Office of Scientific Research, AFOSR-3233 Summary Report, SRI Project 3578. Stanford Research Institute, October.

[45] R. Falcone, & C. Castelfranchi, (2002). From automaticity to autonomy: The frontier of artificial agents. In H. Hexmoor, C. Castelfranchi, & R. Falcone (Ed.), *Agent Autonomy.* (pp. 79-103). Dordrecht, The Netherlands: Kluwer.

[46] G. Ferguson, J. Allen, & B. Miller, (1996). TRAINS-95: Towards a mixed-initiative planning assistant. *Proceedings of the Third Conference on Artificial Intelligence Planning Systems (AIPS-96),* (pp. 70-77). Edinburgh, Scotland,

[47] P. M. Fitts, (Ed.). (1951). *Human Engineering for an Effective Air Navigation and Traffic Control System.* Washington, D.C.: National Research Council.

[48] K. M. Ford, J. M. Bradshaw, J. R. Adams-Webber, & N. M. Agnew, (1993). Knowledge acquisition as a constructive modeling activity. In K. M. Ford & J. M. Bradshaw (Ed.), *Knowledge Acquisition as Modeling.* (pp. 9-32). New York: John Wiley.

[49] K. M. Ford, C. Glymour, & P. Hayes, (1997). Cognitive prostheses. *AI Magazine*, 18(3), 104.

[50] K. M. Ford, C. Glymour, & P. J. Hayes, (Ed.) (1995). *Android Epistemology.* Menlo Park, CA: AAAI Press / The MIT Press.

[51] K. M. Ford, & P. Hayes, (1998). On computational wings: Rethinking the goals of Artificial Intelligence. *Scientific American. Special issue on "Exploring Intelligence"*, 9(4), 78-83.

[52] I. Foster, & C. Kesselman, (Ed.). (1999). *The Grid: Blueprint for a New Computing Infrastructure.* San Francisco, CA: Morgan Kaufmann.

[53] J. Fox, & S. Das, (2000). *Safe and Sound: Artificial Intelligence in Hazardous Applications*. Menlo Park, CA: AAAI Press/The MIT Press.

[54] Y. Gawdiak, J. M. Bradshaw, B. Williams, & H. Thomas, (2000). R2D2 in a softball: The Personal Satellite Assistant. H. Lieberman (Ed.), *Proceedings of the ACM Conference on Intelligent User Interfaces (IUI 2000),* (pp. 125-128). New Orleans, LA, New York: ACM Press,

[55] N. A. Gershenfeld, (1999). *When Things Start to Think*. New York: Henry Holt and Company.

[56] M. AGoodrich, D. R. Olsen Jr., J. W. Crandall, & T. J. Palmer, (2001). Experiments in adjustable autonomy. *Proceedings of the IJCAI_01 Workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents*. Seattle, WA, Menlo Park, CA: AAAI Press,

[57] M. Greaves, H. Holmback, & J. M. Bradshaw, (1999). What is a conversation policy? M. Greaves & J. M. Bradshaw (Ed.), *Proceedings of the Autonomous Agents '99 Workshop on Specifying and Implementing Conversation Policies,* (pp. 1-9). Seattle, WA,

[58] M. Greaves, H. Holmback, & J. M. Bradshaw, (2001). Agent conversation policies. In J. M. Bradshaw (Ed.), *Handbook of Agent Technology.* (pp. in preparation). Cambridge, MA: AAAI Press/The MIT Press.

[59] S. Hamilton, (2001). Thinking outside the box at IHMC. *IEEE Computer*, 61-71.

[60] P. A. Hancock, & S. F. Scallen, (1998). Allocating functions in human-machine systems. In R. Hoffman, M. F. Sherrick, & J. S. Warm (Ed.), *Viewing Psychology as a Whole.* (pp. 509-540). Washington, D.C.: American Psychological Association.

[61] R. Hoffman, P. Feltovich, K. M. Ford, D. D. Woods, G. Klein, & A. Feltovich, (2002). A rose by any other name would probably be given an acronym. *IEEE Intelligent Systems*, July-August, 72-80.

[62] R. R. Hoffman, K. M. Ford, P. J. Hayes, & J. M. Bradshaw, (2003). The Borg hypothesis. *IEEE Intelligent Systems*, in press.

[63] E. Horvitz, (1999). Principles of mixed-initiative user interfaces. *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. Pittsburgh, PA, New York: ACM Press,

[64] E. Horvitz, A. Jacobs, & D. Hovel, (1999). Attention-sensitive alerting. *Proceedings of the Conference on Uncertainty and Artificial Intelligence (UAI '99),* (pp. 305-313). Stockholm, Sweden,

[65] M. Jordan, & M. Atkinson, (1998). *Orthogonal persistence for Java—A mid-term report*. Sun Microsystems Laboratories,

[66] B. Joy, (2000). Why the future doesn't need us. *Wired*, 4, April,

[67] M. Kahn, C. Cicalese, (2001). CoABS Grid Scalability Experiments. O. F. Rana (Ed.), *Second International Workshop on Infrastructure for Scalable Multi-Agent Systems at the Fifth International Conference on Autonomous Agents*. Montreal, CAN, ACM Press, NY,

[68] A. Kay, (1990). User interface: A personal view. In B. Laurel (Ed.), *The Art of Human-Computer Interface Design.* (pp. 191-208). Reading, MA: Addison-Wesley.

[69] G. Knoll, N. Suri, & J. M. Bradshaw, (2001). Path-based security for mobile agents. *Proceedings of the First International Workshop onthe Security of Mobile Multi-Agent Systems (SEMAS-2001) at the Fifth International Conference on Autonomous Agents (Agents 2001),* (pp. 54-60). Montreal, CA, New York: ACM Press,

[70] A. Kolber, (2000). *Defining Business Rules: What Are They Really?* Revision 1.3, Final Report. Business Rules Group (formerly the GUIDE Business Rules Project), July.

[71] N. G. Leveson, (1995). *Safeware: System Safety and Computers: A Guide to Preventing Accidents and Losses Caused by Technology*. Boston, MA: Addison-Wesley.

[72] H. R. Levesque, & R. Brachman, (1985). A fundamental tradeoff in knowledge representation. In R. Brachman & H. R. Levesque (Ed.), *Readings in Knowledge Representation.* (pp. 42-70). San Mateo, CA: Morgan Kaufmann.

[73] J. C. R. Licklider, (1960). Man-computer symbiosis. *IRE Transactions in Electronics. New York: Institute of Radio Engineers.*, 4-11.

[74] H. Lieberman, (Ed.). (2001). *Your Wish is My Command: Programming By Example.* San Francisco, CA: Morgan Kaufmann.

[75] F. Lopez y Lopez, M. Luck, & M. d'Inverno, (2001). A framework for norm-based inter-agent dependence. *Proceedings of the Third Mexican Internation Conference on Computer Science.*

[76] F. Lopez y Lopez, M. Luck, & M. d'Inverno, (2002). Constraining autonomy through norms. *Proceedings of the Conference on Autonomous Agents and Multi-Agent Systems,* (pp. 674-681). Bologna, Italy,

[77] S. Lubar, (1993). *InfoCulture: The Smithsonian Book of Information and Inventions*. Boston, MA: Houghton Mifflin Company.

[78] M. Luck, M. D'Inverno, & S. Munroe, (2002). Autonomy: Variable and generative. In H. Hexmoor, C. Castelfranchi, & R. Falcone (Ed.), *Agent Autonomy.* (pp. 9-22). Dordrecht, The Netherlands: Kluwer.

[79] E. C. Lupu, & M. S. Sloman, (1999). Conflicts in policy-based distributed systems management. *IEEE Transactions on Software Engineering—Special Issue on Inconsistency Management*.

[80] P. Maes, (1997). Agents that reduce work and information overload. In J. M. Bradshaw (Ed.), *Software Agents.* (pp. 145-164). Cambridge, MA: AAAI Press/The MIT Press.

[81] S. Mann, (1997). Wearable computing: A first step toward personal imaging. *IEEE Computer*, 30(2), 25-32.

[82] T. R. Mitrovich, K. M. Ford, & N. Suri, (2001). Transparent redirection of network sockets. *OOPSLA WorkshBp on Experiences with Autonomous Mobile Objects and Agent-based Systems*.

[83] D. Morris, (2002). *Peoplewatching*. London, England: Vintage.

[84] R. R. Murphy, (2000). *Introduction to AI Robotics*. Cambridge, MA: The MIT Press.

[85] D. A. Norman, (1990). The 'problem' with automation: Inappropriate feedback and interaction, not 'over-automation'. In D. E. Broadbend, J. Reason, & A. Baddeley (Ed.), *Human Factors in Hazardous Situations.* (pp. 137-145). Oxford, England: Clarendon Press.

[86] D. A. Norman, (1992). *Turn Signals are the Facial Expressions of Automobiles*. Reading, MA: Addison-Wesley.

[87] D. A. Norman, (1997). How might people interact with agents? In J. M. Bradshaw (Ed.), *Software Agents.* (pp. 49-55). Cambridge, MA: The AAAI Press/The MIT Press.

[88] S. L. Oviatt, P. Cohen, L. Wu, J. Vergo, L. Duncan, B. Suhm, J. Bers, T. Holzman, T. Winograd, J. Landay, J. Larson, & D. Ferro, (2000). Designing the user interface for multimodal speech and pen-based gesture applications: State-of-the-art systems and future research directions. *Human Computer Interaction*, 15(4), 263-322.

[89] A. P. Pentland, (1998). Wearable intelligence. *Scientific American. Special issue on "Exploring Intelligence"*, 9(4), 90-95.

[90] M. Polanyi, (1962). *Personal Knowledge: Toward a Post-Critical Philosophy*. Chicago, IL: The University of Chicago Press.

[91] D. Pynadath, & M. Tambe, (2001). Revisiting Asimov's first law: A response to the call to arms. *Proceedings of ATAL 01*.

[92] N. Sarter, D. D. Woods, & C. E. Billings, (1997). Automation surprises. In G. Salvendy (Ed.), *Handbook of Human factors/Ergonomics, 2nd Edition.* New York, NY: John Wiley.

[93] R. K. Sawyer, (2001). *Creating Conversations: Improvisation in Everyday Discourse*. Cresskill, NJ: Hampton Press.

[94] P. Scerri, D. Pynadath, & M. Tambe, (2002). Adjustable autonomy for the real world. In R. Falcone (Ed.), *Agent Autonomy.* (pp. 163-190). Dordrecht, The Netherlands: Kluwer.

[95] P. Schelde, (1993). *Androids, Humanoids, and Other Science Fiction Monsters*. New York: New York University Press.

[96] D. Schreckenghost, C. Martin, P. Bonasso, D. Kortenkamp, T. Milam, & C. Thronesbery, (2003). Supporting group interaction among humans and autonomous agents. *Submitted for publication*.

[97] D. Schreckenghost, C. Martin, & C. Thronesbery, (2003). Specifying organizational policies and individual preferences for human-software interaction. *Submitted for publication*.

[98] K. E. Seamons, M. Winslet, & T. Yu, (2001). Limiting the disclosure of access control policies during automated trust negotiation. *Proceedings of the Network and Distributed Systems Symposium*.

[99] Y. Shoham, & M. Tennenholtz, (1992). On the synthesis of useful social laws for artificial agent societies. *Proceedings of the Tenth National Conference on Artificial Intelligence,* (pp. 276-281). San Jose, CA.

[100] M. Sierhuis, (2001) *Brahms: A Multi-Agent Modeling and Simulation Language for Work System Analysis and Design*. Doctoral, University of Amsterdam.

[101] M. Sierhuis, J. M. Bradshaw, A. Acquisti, R. Van Hoof, R. Jeffers, & A. Uszok, (2003). Human-agent teamwork and adjustable autonomy in practice. *Proceedings of the Seventh International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*. Nara, Japan.

[102] W. J. Smith, (1977). *The Behavior of Communicating*. Cambridge, MA: Harvard University Press.

[103] T. E. Starner, (1999) *Wearable Computing and Contextual Awareness*. Doctor of Philosophy, Massachusetts Institute of Technology.

[104] D. L. Still, & L. A. Temme, (2001). OZ: A human-centered cockpit display. *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference*. Orlando, FL.

[105] N. Suri, J. M. Bradshaw, M. R. Breedy, P. T. Groth, G. A. Hill, & R. Jeffers, (2000). Strong Mobility and Fine-Grained Resource Control in NOMADS. *Proceedings of the 2nd International Symposium on Agents Systems and Applications and the 4th International Symposium on Mobile Agents (ASA/MA 2000).* Zurich, Switzerland, Berlin: Springer-Verlag.

[106] N. Suri, J. M. Bradshaw, M. R. Breedy, P. T. Groth, G. A. Hill, R. Jeffers, T. R. Mitrovich, B. R. Pouliot, & D. S. Smith, (2000). NOMADS: Toward an environment for strong and safe agent mobility. *Proceedings of Autonomous Agents 2000.* Barcelona, Spain, New York: ACM Press.

[107] N. Suri, J. M. Bradshaw, M. H. Burstein, A. Uszok, B. Benyo, M. R. Breedy, M. Carvalho, D. Diller, P. T. Groth, R. Jeffers, M. Johnson, S. Kulkarni, & J. Lott, (2003). DAML-based policy enforcement for semantic data transformation and filtering in multi-agent systems. *Proceedings of the Autonomous Agents and Multi-Agent Systems Conference (AAMAS 2003).* Melbourne, Australia, New York, NY: ACM Press.

[108] N. Suri, J. M. Bradshaw, M. Carvalho, M. R. Breedy, T. B. Cowin, R. Saavendra, & S. Kulkarni, (2003). Applying agile computing to support efficient and policy-controlled sensor information feeds in the Army Future Combat Systems environment. *Proceedings of the Annual U.S. Army Collaborative Technology Alliance (CTA) Symposium*.

[109] N. Suri, J. M. Bradshaw, M. Carvalho, T. B. Cowin, M. R. Breedy, P. T. Groth, & R. Saavendra, (2003). Agile computing: Bridging the gap between grid computing and ad-hoc peer-to-peer resource sharing. O. F. Rana (Ed.), *Proceedings of the Third International Workshop on Agent-Based Cluster and Grid Computing*. Tokyo, Japan.

[110] N. Suri, J. M. Bradshaw, M. R. Breedy, P. T. Groth, G. A. Hill, R. Jeffers, T. R. Mitrovich, B. R. Pouliot, &D. S Smith, (2000). NOMADS: Toward an environment for strong and safe agent mobility. *Proceedings of Autonomous Agents 2000.* Barcelona, Spain, ACM Press, NY.

[111] N. Suri, M. Carvalho, J. M. Bradshaw, M. R. Breedy, T. B. Cowin, P. T. Groth, R. Saavendra, & A. Uszok, (2003). Mobile code for policy enforcement. *Policy 2003*. Como, Italy.

[112] N. Suri, P. T. Groth, & J. M. Bradshaw, (2001). While You're Away: A system for load-balancing and resource sharing based on mobile agents. R. Buyya, G. Mohay, & P. Roe (Ed.), *Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid*, (pp. 470-473). Brisbane, Australia, Los Alamitos, CA: IEEE Computer Society.

[113] D. Suter, (1986). *Suterisms*. New York, NY: Ballantine Books.

[114] M. Tambe, D. Pynadath, C. Chauvat, A. Das , & G. Kaminka, (2000). Adaptive agent architectures for heterogeneous team members. *Proceedings of the International Conference on Multi-Agent Systems (ICMAS 2000).*

[115] M. Tambe, W. Shen, M. Mataric, D.V. Pynadath, D. Goldberg, P. J. Modi, Z. Qiu, & B. Salemi, (1999). Teamwork in cyberspace: Using TEAMCORE to make agents team-ready. *Proceedings of the AAAI Spring Symposium on Agents in Cyberspace*. Menlo Park, CA, Menlo Park, CA: The AAAI Press,

[116] C. Trevarthen, (1979). Communication and cooperation in early infancy: A description of primary intersubjectivity. In M. Bullowa (Ed.), *Before Speech.* (pp. 321-348). Cambridge, England: Cambridge University Press.

[117] A. Uszok, J. M. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. R. Breedy, L. Bunch, M. Johnson, S. Kulkarni, & J. Lott, (2003). KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. *Proceedings of Policy 2003*. Como, Italy.

[118] A. Vanderbilt, (1952/1963). *Amy Vanderbilt's New Complete Book of Etiquette: The Guide to Gracious Living*. Garden City, NY: Doubleday and Company.

[119] H. Verhagen, (2001). Norms and artificial agents. *Sixth Meeting of the Special Interest Group on Agent-Based Social Simulation, ESPRIT Network of Excellence on Agent-Based Computing.* Amsterdam, Holland, http://abss.cfpm.org/amsterdam-01/abssnorms.pdf,

[120] D. Weld, & O. Etzioni, (1994). The firsts law of robotics: A call to arms. *Proceedings of the National Conference on Artificial Intelligence (AAAI 94)*, (pp. 1042-1047).

[121] E. L. Wiener, (1989). *Human Factors of Advanced Technology, "Glass Cockpit" Transport Aircraft. NASA Contractor Report No. 177528*. NASA Ames Research Center.

[122] M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis , B. Smith, & L. Yu, (2002). Negotiating trust on the Web. *IEEE Internet Computing*, 30-37.