Living With Agents and Liking It: Addressing the Technical and Social Acceptability of Agent Technology

Jeffrey M. Bradshaw*, Maarten Sierhuis, Alessandro Acquisti***, Paul Feltovich*, Robert Hoffman*, Renia Jeffers*, Niranjan Suri*, Andrzej Uszok*, and Ron Van Hoof****

* Institute for Human and Machine Cognition (IHMC), University of West Florida, 40 S. Alcaniz, Pensacola, FL 32501
{jbradshaw, pfeltovich, rhoffman, rjeffers, nsuri, auszok}@ai.uwf.edu
** RIACS and QSS, NASA Ames, MS T35B-1, Moffett Field, CA 94035, {msierhuis, rvanhoof}@mail.arc.nasa.gov
*** SIMS, UC Berkeley, 102 South Hall, Berkeley, CA 94720, acquisti@mail.arc.nasa.gov

Abstract

This paper summarizes our efforts to address some of the technical and social aspects of agent design for increased human acceptability. From a technical perspective, we want to be able to 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 possible that the autonomy of agents can always be bounded by explicit enforceable policy that can be continually adjusted to maximize their effectiveness and safety in both human and computational environments. From a social perspective, we want agents to be designed so as 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. We see these technical and social factors as key to providing the reassurance and trust that are prerequisite to the widespread acceptance of agent technology for non-trivial applications.

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 fiction creatures permeate our culture, forming the unconscious 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* [7]. 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. Despite such innocuous beginnings, the dominant public image of artificially intelligent embodied creatures often has been more a nightmare than a dream. Would the awesome power of robots reverse the master-slave relationship with humans? 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; the more human-like the robot, the more likely to exhibit human frailties and eccentricities. 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!

Figure 1. Powerless in the grasp of a robot (From Astounding Science Fiction, October 1953 [25, p. 383]).

[1] Works by authors such as Schelde [30] 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 [25] provides a useful grand tour of the subject. See Ford, Glymour, and Hayes [16] for a delightful collection of essays on android epistemology.

"Agents occupy a strange place in the realm of technology," summarizes Don Norman, "leading to much fear, fiction, and extravagant claims." 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 in the case of buggy or malicious agents. Because ever more powerful intelligent agents will increasingly differ in important ways from conventional software of the past, we need to take into account the 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.
>
> The social part of acceptability is to provide reassurance that all is working according to plan... This is [also] a non-trivial task. [29, p. 51].

This paper summarizes our efforts to address some of the technical and social aspects of agent design for increased human acceptability. From a technical perspective, we want to be able to 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 possible that the autonomy of agents can always be bounded by explicit enforceable policy that can be continually adjusted to maximize their effectiveness and safety in both human and computational environments.

From a social perspective, we want agents to be designed so as 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 prerequisite to the widespread acceptance of agent technology for non-trivial applications.

Technical Aspects of Agent Acceptability

While agent systems of the past have often been built on fragile experimental platforms, examples of the kinds of basic infrastructure that will be required for robust agent systems 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 ([17]; 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 [19; 37].

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 2). 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 [35; 36]) 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.

Here 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. 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.

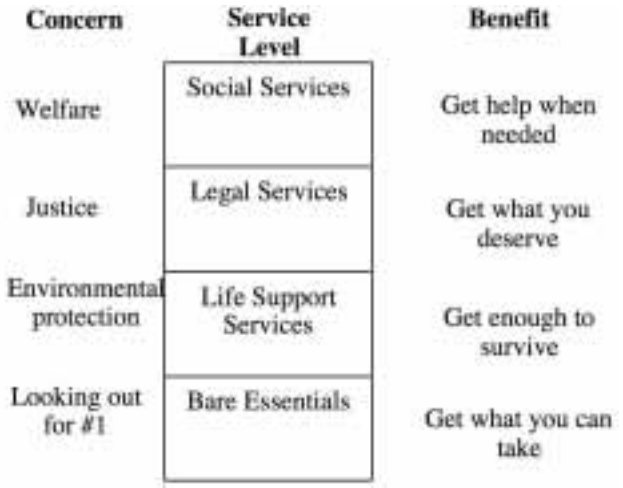

Figure 2. Required elements of future infrastructure for agent technology.

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 [35; 36]. 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 denial-of- service attacks;
- protection of agent execution state, even in the face of unanticipated system failure.

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 standard Java enforcement capabilities described above, but also supports access revocation under all circumstances, dynamic resource control and full state capture on demand for any Java agent or service.

Protection of agent 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. 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. Finally, the availability of resource control mechanisms in the environment simplifies the task of developing systems for resource-constrained situations such as those often encountered in space and military applications. We are working with Sun Microsystems Laboratories to help incorporate resource control capabilities into commercial Java Virtual Machines.

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, 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 [23].

In contrast to commercial Java VM's, the Aroma VM has been enhanced with the capability to capture the execution state of any running 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.

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 [38]. 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) [28], 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.

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. 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.

Policy representation and reasoning. KAoS policy and domain services rely on a DAML description-logic-based ontology of the computational environment, application context, and the policies themselves that enables runtime extensibility and adaptability of the system, as well as the ability to analyze policies relating to entities described at different levels of abstraction [6; 40]. The representation facilitates careful reasoning about policy disclosure, conflict detection, and harmonization, and about domain structure and concepts.2 A graphical interface hides the complexity of this representation and reasoning from the user.

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.

A policy is a statement enabling or constraining execution of some type of action by one or more actors in relation to various aspects of some situation. Our current policy ontology distinguishes 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) [12].

In DAML, a policy is represented as an instance of the appropriate policy type with associated values for properties: priority, update time stamp and a site of enforcement. 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).

In the current version of KAoS, changes or additions to policies in force or a change in status of an actor (e.g., a human administrator being given new permissions; software 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 [26].

Figure 3 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 very different levels of abstraction. 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.

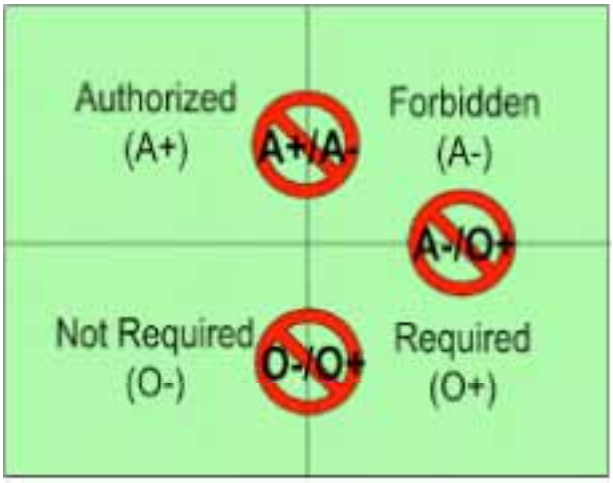

Figure 3. Types of policy conflicts

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. 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 [5]. These cornerstones of human-agent interaction can be summarized as follows:

- 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 but rather how tasks can best be shared and performed by both humans and agents working in concert [15; 21].
- Approaches to adjustable autonomy are a useful means to enable principled yet dynamic flexibility in the roles of people and agents [13]. 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 [20].
- Maintaining appropriate mutual awareness among team members is paramount for human-agent

interaction in complex environments [8; 14]. 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 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.

- In designing human-agent systems, it is important to dispel the misconception that automation is a simple substitution of machine for human effort [8; 42]. 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.

Teamwork has become the most widely-accepted metaphor for describing the nature of multi-agent cooperation. In most approaches, the key concept is that of shared knowledge, goals, and intentions that function as the glue that binds team members together [11]. 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, there is a growing interest in various dimensions of human-agent interaction [4]. Unlike autonomous systems designed primarily to take humans out of the loop, many new efforts are specifically motivated by the need to support close human-agent interaction [18; 22; 24; 39].

Under NASA sponsorship, we are investigating issues in human-robotic teamwork and adjustable autonomy. Future human missions to the Moon and to Mars will undoubtedly need the increased capabilities for human-robot collaborations we envision. Astronauts will live, work, and perform laboratory experiments in collaboration with robots inside and outside their spacecraft and habitats on planetary surfaces.

An adequate approach to design of cooperative autonomous systems requires first and foremost a detailed understanding of how people actually work. To this end, researchers at RIACS have developed Brahms, an agent-based design toolkit that can be used to model and simulate realistic work situations in space [10; 33]. The ultimate objective is to produce agent-based simulations in Brahms that could form the basis for the design of robotic and software agent functions for actual operations. On its part, IHMC is enhancing the KAoS and Nomads agent frameworks to incorporate explicit general models of teamwork, adjustable autonomy, notification, mobility, and resource control appropriate for space operations scenarios. The general models are represented largely in the form of policies. Through a Brahms-based study of astronaut work practices, an appreciation of relevant past research from the social sciences, and the application of policy-based technical mechanisms in KAoS and Nomads, we aim to improve the social aspects of agent acceptability.

Brahms

Brahms is a language and modeling and simulation environment based on the idea of "situated action" [9; 34] and offers to the researcher a tool to represent and study the richness of activity theory and "work practice" [10]. 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 model how work and information actually flow, nor can they properly design software agents that help automate human tasks or interact with people as their collaborators. For these goals, what is needed is a model that includes aspects of reasoning found in an information-processing model, plus aspects of geography, agent movement, and physical changes to the environment found in a multi-agent simulation - interruptions, coordination, impasses, and so on. A model of work practice focuses on informal, circumstantial, and located behavior by which synchronization occurs (such that the task contributions of humans and machines flow together to accomplish goals) and allows the researcher to capture (at least part of) the richness of activity theory.

Brahms is based on an Agent-Oriented Language (AOL) with a well-defined syntax and semantics. A Brahms model can be used to simulate human-machine systems, for what-if experiments, for training, "user models," or driving intelligent assistants and robots. For a full description of Brahms, the reader is referred to [33]. The run-time component-the Brahms virtual machine-can execute a Brahms model, also referred to as a simulation run.

The Brahms architecture is organized around the following representational constructs:

Groups of groups containing
Agents who are located and have
Beliefs that lead them to engage in
Activities specified by
Workframes
Workframes consist of
Preconditions of beliefs that lead to
Actions, consisting of
Communication Actions
Movement actions
Primitive Actions
Other composite activities
Consequences of new beliefs and facts
Thoughtframes that consist of
Preconditions and
Consequences

Brahms relates knowledge-based models of cognition (e.g., task models) with discrete simulation and the behavior-based subsumption architecture. In Brahms, agents' behaviors are organized into activities, inherited from groups to which agents belong. Most importantly, activities locate behaviors of people and their tools in time and space, such that resource availability and informal human participation can be taken into account. A model of activities doesn't necessarily describe the intricate details of reasoning or calculation, but instead captures aspects of the social-physical context in which reasoning occurs. Thus Brahms differs from other simulation systems by incorporating the following:

- Activities of multiple agents located in time and space;
- Conversations;
- Descriptions of how information is represented, transformed, reinterpreted in various physical modalities.

Studying Work Practice on the International Space Station (ISS)

Consistent with our emphasis in understanding teamwork in practice, our study necessarily begins with a detailed real-world understanding of how the astronauts actually work onboard the ISS. Our first step has been to develop a Brahms model of the daily work practice of the crew. Secondly, we would extend that model to include various PSA and Robonaut use scenarios. Finally, to support real-time operational teamwork capabilities for these robots, we would enhance the Brahms model and software to run in conjunction with KAoS policies and agent services, and NOMADS capabilities.

The objective of studying crew work practice is threefold: 1) to study how actual crew work practices emerge from planned activities and written procedures; 2) to discover opportunities for collaborative robots such as the Personal Satellite Assistant (PSA) [18] and Robonaut [2] to assist with the crew's work; and 3) to learn more about how teamwork actually occurs in practice. A more complete account of the Brahms ISS modeling effort to date may be found in [1].

From a modeling perspective, the primary challenge we have addressed is how to represent the schedule and the practice of following a schedule. Work practice might diverge from a procedure in several ways:

- Work practices specific to certain procedures might emerge. That is, the crew has learned how to perform the work, and what they do is not documented (yet) in the procedure. By observations of videos, debriefs, and interviews, we can note that certain activities are regularly executed in a particular way, possibly not the exact way described in the procedure.
- Other events not described in the procedures might occur: interruptions, delays (such as the time spent looking for tools), and so on. These events could be defined as work practice not specific to any particular activity (e.g., a preferred location that a crew member uses for storing the tools needed for a task).
- Errors and failures might arise. These might have well-defined statistical properties that we can observe from data and insert in the model.
- In addition to this, we are currently working on the meta-level of *"just-in-time replanning activity,"* where the simulated agent, noting that a certain activity scheduled for a certain time and length is taking more (or less) than the time allocated for it, has to decide in real time the next step in his execution plan, which might include contacting mission controllers for advice.

Flashlight Scenario

The detail in the Brahms model has helped us notice unexpected opportunities for robotic assistance. For example, a simple distraction like one crewmember having to stop and look for missing tools can tie up valuable human resources. Designing features to support mundane activities that waste valuable crew time into the PSA is seen as a high priority.

In certain areas of the International Space Station, such as space station module connections or spaces behind panels, astronauts have to work in near darkness. Given that maintenance is often required there, flashlights become a necessary tool (figure 4). A simple but useful task for the PSA would be for it to aim its light wherever an astronaut needs it, rather than requiring one of the astronauts to hold a flashlight.

As a simple but relevant first step to understanding more complex situations, we are analyzing a flashlight scenario through modeling in KAoS and Brahms. The analysis involves identifying the goals, teamwork steps, and collaborative activities as they unfold. The modeling involves coupling a detailed executable Brahms scenario description to KAoS policies for various aspects of human-robotic interaction (e.g., teamwork, notification, and adjustable autonomy). Even in such a seemingly simple scenario, we are finding incredible grist for understanding the complexities of human-agent interaction. Beyond the

understanding we are gleaning from this first step, a measure of our success will be the degree to which we are able to effectively reuse portions of the Brahms models and KAoS policies in other scenarios in ways that will lead to generic models of human-agent interaction, making development of socially-wise autonomous systems more rapid and reliable.

Figure 4. Still image from a video of an astronaut performing a task similar to our flashlight scenario.

In creating teamwork and adjustable autonomy policies, we seek to incorporate the best of previous research on human-centered collaboration and teamwork, while simultaneously grounding such research in our own work practice study experience. In addition to surveying studies of multi-agent teamwork and human-centered perspectives cited above, we are assessing the contributions of allied fields ranging from cognitive function analysis [3], to studies of animal signaling and cooperation, the roles of values and affect, frameworks for context-sensitive notification and alerts [31; 32], and the enablers of effective delegation in humans [27]. We have even consulted books of etiquette [41]!

From our preliminary studies of human teamwork in practice to date we have realized that the longstanding emphasis in teamwork research on "joint goals" and fixed task-specific "roles" seems to have overshadowed important aspects of teamwork in practice such as shared context, awareness, identity, and history. Unlike the relatively rigid joint intentions of typical agent teamwork models, experience in work practice underscores the importance of conceptualizing agreements as things that are forever tentative and subject to ongoing negotiation. An interesting finding has been that some of the most important settings for the coordination of teamwork are not in formal planning meetings but around the breakfast table. Such settings can also be a time for sharing knowledge gleaned from practice.

During the workshop we anticipate sharing specific results from our work on the flashlight scenario, including the kinds of policies with which we have experimented.

Summary

As we continue our efforts to understand and address the technical and social aspects of agent acceptability over the long term, we hope our work will benefit those who will someday find themselves working closely and continuously with agents over long periods of time, as well as those of us who are interested in making the design of effective agent assistants to humans more efficient and robust.

References

[1] Acquisti, A., Sierhuis, M., Clancey, W. J., & Bradshaw, J. M. (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] Ambrose, R., Culbert, C., & Rehnmark, F. (2001). An experimental investigation of dexterous robots using EVA tools and interfaces. *AIAA, 4593*.

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

[4] Bradshaw, J. M., Beautement, P., Raj, A., Johnson, M., Kulkarni, S., & Suri, N. (2002). Making agents acceptable to people. In N. Zhong & J. Liu (Ed.), *Handbook of Intelligent Information Technology*. (pp. in preparation). Amsterdam, The Netherlands: IOS Press.

[5] Bradshaw, J. M., Sierhuis, M., Acquisti, A., Feltovich, P., Hoffman, R., Jeffers, R., Prescott, D., Suri, N., Uszok, A., & Van Hoof, R. (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.

[6] Bradshaw, J. M., Uszok, A., Jeffers, R., Suri, N., Hayes, P., Burstein, M. H., Acquisti, A., Benyo, B., Breedy, M. R., Carvalho, M., Diller, D., Johnson, M., Kulkarni, S., Lott, J., Sierhuis, M., & Van Hoof, R. (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,

[7] Capek, K. (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.

[8] Christofferson, K., & Woods, D. D. (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.

[9] Clancey, W. J. (1997). *Situated Cognition: On Human Knowledge and Computer Representations*. Cambridge, England: Cambridget University Press.

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

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

[12] Damianou, N., Dulay, N., Lupu, E. C., & Sloman, M. S. (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.

[13] Dorais, G., Bonasso, R. P., Kortenkamp, D., Pell, B., & Schrekenghost, D. (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,

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

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

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

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

[18] Gawdiak, Y., Bradshaw, J. M., Williams, B., & Thomas, H. (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,

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

[20] Hancock, P. A., & Scallen, S. F. (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 Psych. Association.

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

[22] Horvitz, E. (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,

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

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

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

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

[27] Milewski, A. E., & Lewis, S. H. (1994). *Design of intelligent agent user interfaces: Delegation issues.* AT&T Corporate Information Technologies Services Advanced Technology Planning, October 20.

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

[29] Norman, D. A. (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.

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

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

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

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

[34] Suchman, L. A. (1987). *Plans and Situated Action: The Problem of Human-Machine Communication*. Cambridge, MA: Cambridge University Press.

[35] Suri, N., Bradshaw, J. M., Breedy, M. R., Groth, P. T., Hill, G. A., & Jeffers, R. (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,

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

[37] Suri, N., Bradshaw, J. M., Carvalho, M., Cowin, T. B., Breedy, M. R., Groth, P. T., & Saavendra, R. (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,

[38] Suri, N., Groth, P. T., & Bradshaw, J. M. (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,

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

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

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

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