From STEAM to Machinetta:
The evolution of a BDI teamwork model

Nathan Schurr, Rajiv Maheswaran, Paul Scerri, and Milind Tambe

Computer Science Department
University of Southern California
941 W. 37th Place
Los Angeles, CA 90089-0781

Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

schurr@usc.edu, maheswar@usc.edu, pscerri@cs.cmu.edu, tambe@usc.edu

Draft: 1.31.04

1. INTRODUCTION

For applications involving heterogeneous agents working together to achieve complex goals, *teamwork* [1,2,3] has emerged as the dominant coordination paradigm. For domains as diverse as rescue response, military, space, sports and collaboration between human workmates, flexible, dynamic coordination between cooperative agents needs to be achieved despite complex, uncertain, hostile environments. There is now emerging consensus in the multiagent arena that for flexible teamwork among agents, each team member is provided with an explicit model of teamwork, which entails their commitments and responsibilities as a team member. This explicit modeling allows the coordination to be robust, despite individual failures and unpredictably changing environments.

Building on the well developed theory of *joint intentions*[12], STEAM[3] and operationalized teamwork as a set of domain independent rules that describe how teams should work together. The domain independent teamwork rules have been successfully applied to a great variety of domains. From combat air missions[13] to robot soccer[8] to teams supporting human organizations[4] to rescue response[7], applying the same set of rules has resulted in successful coordination between heterogeneous agents. The successful use of the same set of rules in a wide variety of diverse domains provides compelling evidence that it is the principles of teamwork, rather than exploitation of specific domain phenomena, that underlies the success of teamwork based approaches.

Since the same rules can be successfully used in a range of domains, it is desirable to build a reusable software package that encapsulates those rules, while avoiding the cost of implementing the rules for each new domain. The emerging standard for deploying such a package is via *proxies*[4,8]. Each proxy works closely with a single domain agent, representing that agent in the team. We are currently working with the second generation of teamwork proxies, called Machinetta [5,8]. The Machinetta proxies are more lightweight and flexible than the TEAMCORE[4] proxies they have superseded.

While approaches to team work have been shown to be effective for agent teams, new emerging domains of teamwork require agent-human interactions in teams. These emerging domains and the teams that we are developing for them introduce a new set of issues and obstacles. Two algorithms that need to be revised in particular for these complex domains are the algorithms that govern the way the team coordinates and coordinates and the algorithm for allocating roles in the team. Since teamwork is known to have high computational complexity, heuristics are required. For instance, heuristic may work well for coordination in a certain situation, but once the environment has changed a new heuristic would result in much better performance. However, we can leverage the diverse expertise of agents and humans via *adjustable*

autonomy. By representing all coordination tasks as explicit roles, we can give responsibility for key coordination decisions to the appropriate team member by using *transfer-of-control strategies*. Furthermore, we can use predefined sequences of transfer of control actions to take back authority for a decision, if the entity chosen by the adjustable autonomy decision does not respond in a timely manner.

In order to allocate a dynamically changing set of roles to team members, previous mechanisms required too much computation and/or communication and did not handle rapidly changing situations well for teams with many members. We have developed a novel algorithm for role allocation in very large teams in rapidly changing contexts. The algorithm is based on distributed constraint optimization but has several key extensions to reduce communication and deal better with changing circumstances.

2. DOMAINS

Our proxy approach has been applied earlier to several domains such as battlefield simulations and RoboCup soccer simulations [1,4]. Here, we will describe four additional domains that we have used to explore proxy-based teamwork. In each of these domains the same teamwork approach has been applied and been shown to be effective without changes to the key ideas. These proxy-based multiagent teams could potentially help human organizations in rapidly responding to disasters such as earthquakes, fires, or large-scale terrorist attacks.

The first domain is an agent-facilitated human organization. Individual software agents embedded within the organization would act as enduring personal assistants to human users in the organization; more importantly, they would work together in teams in service of cooperative tasks. Such agentified organizations could potentially revolutionize the way a variety of tasks are carried out by human organizations. In an earlier research project called "Electric Elves", an agent system was deployed at USC with a small number of users and ran it continuously for nine months (see Figure 1) [11,6]. The longest running multiagent system in the world, it provided personal assistant agents (proxies) for about a dozen researchers and students here at USC and integrated several schedulers and information agents. The resulting small-scale team of 15-20 agents helped us in daily tasks such as

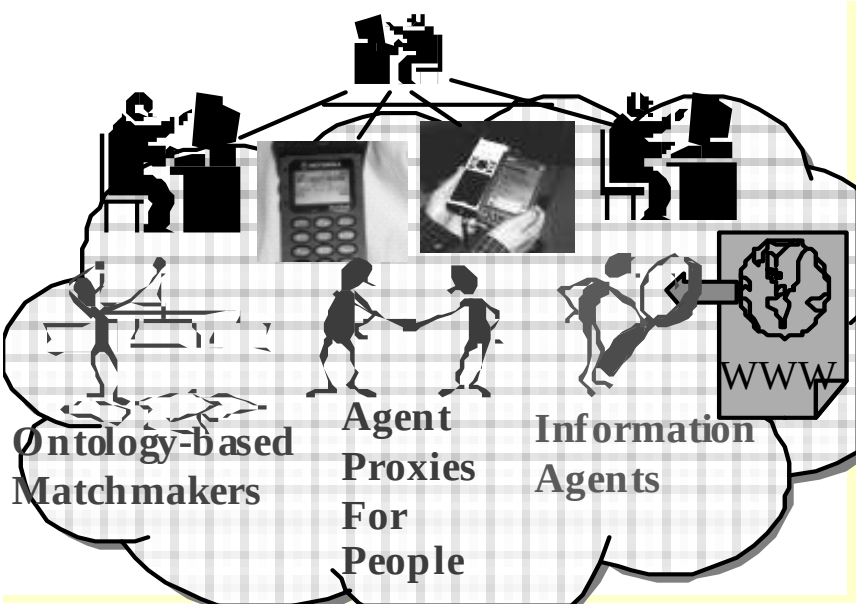

Figure 1

rescheduling meetings, deciding presenters for research meetings, tracking people and ordering meals. Communicating with our palm pilots and cell phones, the personal assistants would attempt to adjust their autonomy appropriately. Building on this experience, we have begun work on a more comprehensive joint project with SRI that is known as CALO [13] (our fourth domain).

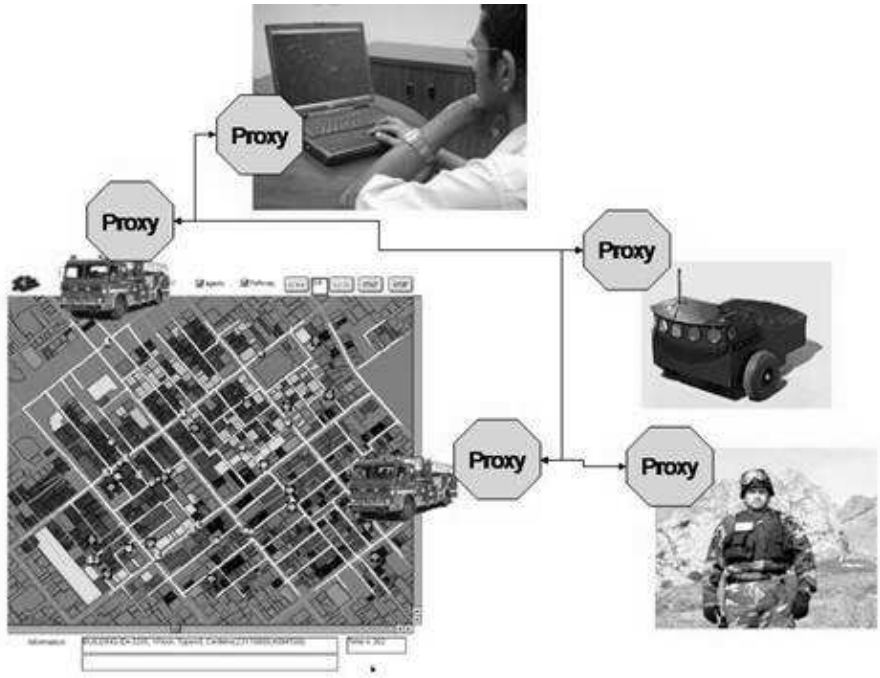

Figure 2: Machinetta Proxies in Disaster Response

In our second domain, disaster response (see Figure 2), we create teams that leverage the unique capabilities of Robots, Agents and People (RAPs). Proxy facilitated teamwork is vital to effective creation of RAP teams. A major challenge stems from the fact that RAP entities may have differing social abilities and hence differing abilities to coordinate with their teammates. In order to fully model these challenges, our experimental platform in this joint project [7] is an extension of the RoboCup Rescue simulation environment [8] that enables human-robot interactions. Fire brigade agents act in a virtual city, while human and robot team members act in the physical world. The fire brigades can search the city after an earthquake has hit and can extinguish any fires that are found. The agents try to allocate themselves to fires in a distributed manner, but can call on the expertise of the human fire chief if required. The fire chief can allocate trucks to fires easily both because of a more global view of the situation and because the spatial, high-level reasoning required is well suited to human capabilities. Thus, the fire chief's proxy must carefully adjust its own autonomy when accepting and rejecting roles.

A third domain that we have applied the proxies to is a sensor network environment. Our previous work dealt with stationary distributed hardware sensors [9] and our more recent work focuses on mobile sensors. The proxies will eventually be migrated onto real robots and hardware. As a first step, we have developed a simulator which encompasses a sensor network that is to be maintained over an extended period of time. This simulator has allowed us to conduct experiments with large scale teams.

In our fourth domain, we recently, in conjunction with SRI International, have undertaken an ambitious project titled CALO, whose aim is to create a wide-ranging and functional personal assistant agent that maintains a persistent presence by continuously learning from its user and acting on their user's behalf. Among the many organizational domains that have been identified as key areas for development are automated meeting scheduling, calendar administration, data handling for aggregate purchasing, and project management. For most applications, agents cannot operate in isolation as performance often depends on the successful coordination of multiple members (and their agent proxies) relevant to the task. Designing a ubiquitous agent that propagates its utilization by providing incentives to both institutions and individuals critically depends on the development efficient and effective teamwork.

3. PROXIES

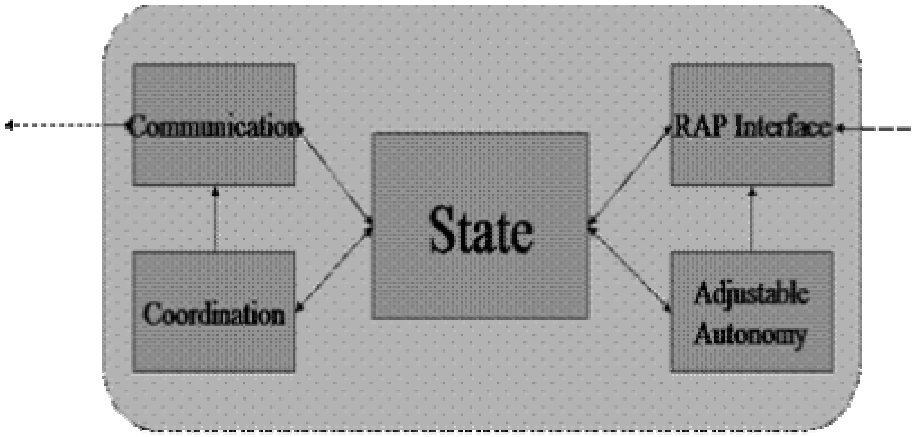

Figure 3. Machinetta Proxy Architecture

Proxies are pieces of software that facilitate the actions and communication necessary to work cooperatively on a team plan. In this section, we will describe the inner workings of a Machinetta proxy. Machinetta proxies are lightweight, domain-independent Java programs, capable of performing the activities required to get a large group heterogeneous entities to work together. The proxies are designed to run on a number of platforms including laptops, robots and handheld devices. A proxy's software is made up of five components as seen in Figure 3.

Communication: communication with other proxies
Coordination: reasoning about team plans and communication
State: the working memory of the proxy
Adjustable Autonomy: reasoning about whether to act autonomously or pass control to the team member
RAP Interface: communication with the team member

Each component abstracts away details allowing other components to work without considering those details. The *RAP interface* component is the only part of the proxy that needs to be designed for a specific type of team member. For example, the *RAP interface* for a person playing the role of fire chief in the disaster rescue domain is a large graphical interface, while for agents a simple socket communicating a small, fixed set of messages is sufficient. This design has allowed the Machinetta to scale up to run 200 proxies on two desktop computers [14].

3.1 BDI Teamwork

A team of proxies implements *Team Oriented Plans* (TOPs). A TOP is a team-level description of the activities that need to be performed in order to achieve the goals of the team. It consists of reactive team plans, TOP specifies roles, relationships between roles, conditions for initiating a plan and terminating a plan. The proxies dynamically instantiate plans when, during the course of execution, their current states match a plan's required trigger conditions. The proxy communication policy [4] determines precisely which messages should be sent among proxies to ensure that cohesion is maintained.

In developing Machinetta, we have focused on the joint intentions theory [12], due to its detailed formal specification and prescriptive power. The joint intentions framework [12] focuses on a team's mental state, called a joint intention. A team intends a team action if team members are jointly committed to completing that team action, while mutually believing that they were doing it. A joint commitment in turn is defined as a joint persistent goal (JPG). The team Θ's JPG is to achieve p, where p stands for completion of a team

action, is denoted JPG(Θ, p, q). The variable q is an irrelevance clause, as described below. It enables a team to drop the JPG should they mutually believe q to be false. JPG(Θ, p, q) holds if three conditions are satisfied:

1. All team members mutually believe that p is currently false.
2. All team members have p as their mutual goal, i.e., they mutually know that they want p to be true eventually.
3. All team members mutually believe that until p is mutually known to be achieved, unachievable or irrelevant, they mutually believe that they each hold p as a weak goal (WAG).
4. WAG(μ, p, Θ, q), where μ is a team member in Θ, implies that one of the following holds:
    - μ believes p is currently false and wants it to be eventually true, i.e., p is a normal achievement goal); or
    - Having privately discovered p to be achieved, unachievable or irrelevant (because q is false), μ has committed to having this private belief become Θ 's mutual belief.

To enter into a joint commitment (JPG) in the first place, all team members must establish appropriate mutual beliefs and commitments. The commitment to attain mutual belief in the termination of p is a key aspect of a JPG. This commitment ensures that team members stay updated about the status of team activities, and thus do not unnecessarily face risks or waste their time.

The Machinetta proxies dynamically instantiate plans when, during the course of execution, their current states match a plan's required trigger conditions. This team plan belief can then trigger domain-specific behavior through the interface with a proxy's specific RAP. Upon successfully triggering a new plan, the proxies perform the “establishJointCommitment” (as described above) procedure specified by their coordination policy to ensure that all proxies agree on the plan. Because each proxy maintains separate beliefs about these joint goals, the team can detect (in a distributed manner) any inconsistencies among team members' plan beliefs. The proxies then use termination conditions, specified in the TOP, as the basis for automatically generating the communication necessary to jointly terminate a team plan when those conditions are met.

Roles are slots for specialized execution that the team may potentially fill at runtime. Upon instantiation of a newly triggered plan, the proxies also instantiate any associated roles. The initial plan specification may name particular team members to fill these roles, but often the roles are unfilled, which are then subject to role allocation. The proxies themselves have no ability to achieve goals at the domain level; instead, they must ensure that all of the requisite domain-level capabilities are brought to bear by informing team members of their responsibility to perform instantiated roles that are allocated to them.

For example, there is a TOP that describes the team plan for a group of researchers to give a presentation together. Once the precondition that the presentation is scheduled has been met, the plan becomes active and the proxies begin establishing the joint commitment to the new plan. Roles are then taken on by team members that have the joint commitment to the new plan. If one of the team members finds out that the presentation has been cancelled, this termination condition is communicated to all those that have the joint commitment. This enables the team to jointly terminate the plan rather than leave some unaware researchers to still prepare for their roles in the cancelled presentation.

3.2 Adjustable Autonomy

Adjustable autonomy reasoning determines whether the proxy will act autonomously or attempt to get input on a coordination decision from a team member. Such reasoning is critical to the successful deployment of heterogeneous teams containing people.

We have developed a flexible approach to adjustable autonomy, building on the notion of transfer-of-control strategies [[10]]. Transfer-of-control strategies are pre-planned sequences of actions to either transfer control or change team plans to buy more decision making time. The strategies are executed until the team member in control makes the decision. This flexible back and forth of control is critical in domains where decisions must be made in a timely manner but the team member may not be available to make a decision.

Human reasoning can be far superior to proxy reasoning. However, this does not hold for all decisions. For this reason, Machinetta adjustable autonomy reasoning is invoked on each proxy decision. To make this reasoning easier to perform, all of the routine coordination tasks that the proxies perform are explicitly represented as roles [7]. For example, determining the priority of a team plan is an explicit role, as is allocation of a role within a plan. Explicit representation of coordination tasks as roles makes it much easier to invoke adjustable autonomy reasoning and to pass control to a team member to make a decision. In practice, an overwhelming majority of coordination roles are completed by the proxies, but in key situations, human expertise can be brought to bear. In the future, we envision coordination roles also being assigned to specially designed agents that have resources available to perform very high quality coordination reasoning.

3.3 Role Allocation

To allocate unfilled roles to team members, we have developed a novel role allocation algorithm based on ideas from distributed constraint optimization problems (DCOP) [10]. Based on valued constraints, DCOP is a powerful and natural representation for the role allocation problem. Mapping the problem to a well-known paradigm like DCOP allows us to leverage a large body of work for the design of the algorithm. However, while DCOP-based algorithms have been previously applied to limited role allocation problems, they have several shortcomings when used for very large teams in dynamic environments.

Our DCOP-based role allocation algorithm for teams, Low communication Approximate DCOP (LA-DCOP), uses a representation where agents are variables that can take on values from a common pool of roles to be assigned. LA-DCOP is a specialized algorithm that has been designed for this role allocation application. The mechanism for allocating values to variables encapsulates three novel ideas. First, *tokens* are used to regulate access to values. Only the agent currently holding the token for a particular value can consider assigning that value to its variable. The use of tokens removes the possibility of several agents taking on the same role, thus dramatically reducing the need to communicate about and repair conflicts. Second, probabilistic information is used by the proxies to estimate the likely characteristics of optimal allocations. This information is used to quickly guide the search towards good solutions. In particular, the agents calculate a minimum *threshold* on the expected capability of the agent that will be assigned to a role in a good allocation. If the agent's capability to perform the role is less than the threshold capability, it does not consider taking on the role. Finally, to deal with groups of constrained roles, we introduce the idea of allowing variables to have *potential tokens*. When groups of roles are constrained, instead of committing to a role by assigning the value represented by a token, a team member accepts a potential token indicating that it will accept the role if and only if all other constrained roles can also be filled. While team members are being found to fill the other constrained roles, a team member with a potential token can perform other roles. Only when team members have been found for all roles will the members of the newly formed coalition actually take on the roles. Multiple team members may accept a single potential token, once again exploiting probabilistic information to determine if they will drop a potential token upon the arrival of a different real token.

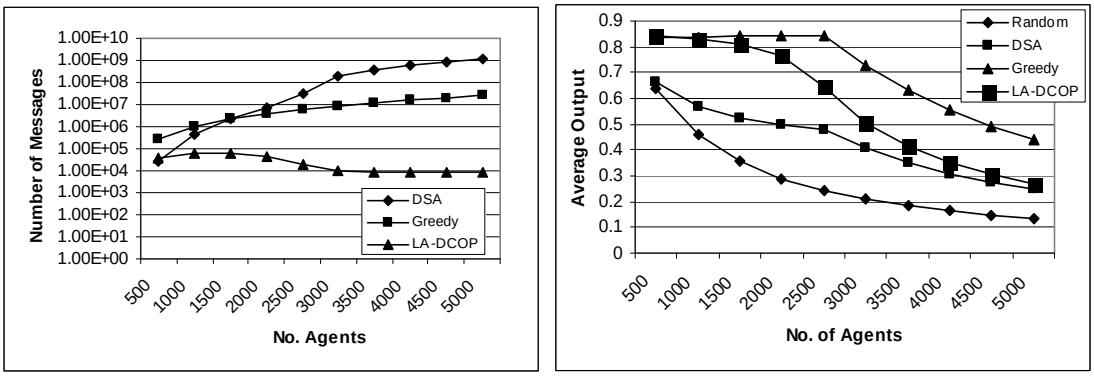

Figure 4. LA-DCOP Comparison

Experiments, on an abstract role allocation simulator, show that our approach is able to give similar results over a wide range of working conditions, while keeping very low communication overhead. Figure 4 shows a comparison of our current role allocation algorithms to that of DSA [[11]], a state of the art competing approximate algorithm for DCOP. DSA is a published algorithm for obtaining fast, approximate solutions for DCOP. DSA has been shown to outperform other DCOP algorithms when optimal solutions are not required. The results show Machinetta outperforming DSA in two ways: Machinetta uses far fewer messages, and also provides increased team output, which we use as a measure of quality of a solution.

4. SUMMARY

This chapter reports on Machinetta, a proxy-based approach to enabling teamwork among diverse entities. The Machinetta proxies equip each team member with a model of the commitments and responsibilities necessary for teamwork. This model is derived from a BDI framework and the notion of joint intentions. These proxies have been effectively applied to a wide range of domains from personal assistants to disaster rescue. Furthermore, these advancements have all been packaged into a lightweight Java process. Two key aspects of these proxies are the algorithms implemented in role allocation and adjustable autonomy. These Machinetta proxies apply the design and analysis of a new role allocation algorithm (LA-DCOP). In addition, adjustable autonomy is integrated into the proxies by representing coordination tasks as domain level tasks. The approach outlined above has allowed for the construction of proxies that have repeatedly and definitively demonstrated effective teamwork in diverse domains.

REFERENCES

[1] N. Jennings. The archon systems and its applications. Project Report, 1995.

[2] Yen, J., Yin, J., Ioerger, T., Miller, M., Xu, D., and Volz, R. CAST: Collaborative Agents for Simulating Teamwork. IJCAI, pages 1135-1144, 2001

[3] Tambe, M. Agent architectures for flexible, practical teamwork. *National Conference on AI* (AAAI97), pages 22-28, 1997.

[4] Pynadath, D and Tambe, M. An automated teamwork infrastructure for heterogeneous software agents and humans. *Journal of Autonomous Agents and Multi-Agent Systems, Special Issue on Infrastructure and Requirements for Building Research Grade Multi-Agent Systems*, pages 7:71-100, 2003.

[5] http://teamcore.usc.edu/doc/Machinetta

[6] Chalupsky, H, Gil, Y., Knoblock, C., Lerman, K., Oh, J., Pynadath, D., Russ, T., Tambe, M. Electric Elves: Agent Technology for Supporting Human Organizations. *AI Magazine*

[7] P. Scerri, D. V. Pynadath, L. Johnson, Rosenbloom P., N. Schurr, M Si, and M. Tambe. A prototype infrastructure for distributed robot-agent-person teams. *The Second International Joint Conference on Autonomous Agents and Multiagent Systems*, 2003.

[8] Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., and Matsubara, H. RoboCup: A challenge problem for AI. *AI Magazine*, 18(1):73-85, Spring 1997.

[9] Modi, P., Shen, W., Tambe, M., Yokoo, M. An asynchronous complete method for distributed constraint optimization. *Proceedings of the second International Joint conference on agents and multiagent systems* (AAMAS), 2003

[10] Scerri, P., Pynadath, D., and Tambe, M. Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research*, 17:171{228, 2002.

[11] Fitzpatrick, S. and Meertens, L. Soft, Real-Time, Distributed Graph Coloring using Decentralized, Synchronous, Stochastic, Iterative-Repair, Anytime Algorithms: A Framework. *Kestrel Institute Technical Report* KES.U.01.5., May 2001

[12] Cohen, P. R., and Levesque, H. J. 1991. Teamwork. *Nous* 35.

[13] Hill, R., Chen, J., Gratch, J., Rosenbloom, P., and Tambe, M., 1997 Intelligent agents for the synthetic battlefield: A company of rotary wing aircraft. *Innovative Applications of Artificial Intelligence (IAAI-97)*

[13] http://www.ai.sri.com/project/CALO

[14] Paul Scerri, Elizabeth Liao, Guoming Lai, Katia Sycara, Yang Xu, Mike Lewis. Coordinating Very Large Groups of Wide Area Search Munitions. Book Chapter.