Facilitating Human Collaboration with Agents

James E. Just
Global InfoTek, Inc.
jjust@globalinfotek.com

Mark R. Cornwell
Global InfoTek, Inc.
mcornwell@globalinfotek.com

Michael N. Huhns
University of South Carolina
huhns@sc.edu

Abstract

Ad hoc cross-agency teams are often needed to deal with actual, imminent, or potential crises that involve multiple geographic or political jurisdictions or require coordinated expertise from organizations with different responsibilities. Many of these teams are wholly or partially virtual. We have implemented an initial set of personal and organizational agents, agent-based work flows, and Web services that facilitates the establishment of collaborating teams of humans who work for different organizations and whose primary mode of interaction is virtual. The agents augment existing email and office automation applications in an organization. Preliminary results show the potential for a reduction of two orders of magnitude in the time needed to form a team. The system is being enhanced to manage on-going collaborations of team members and workgroups.

1 Introduction

Imagine composing a team that must response to a potential or imminent crisis. In these times, such a team might involve national, state and local police and policy-makers, national intelligence assets, specially trained military units, and other first responders. There are two general problem areas: (1) how to form such a team efficiently from these disparate organizations and how to get the new team members to work together effectively.

Forming an effective cross-agency team can be a significant bottleneck. Many of the traditional approaches to teams, such as assigning primary responsibility to one agency and having liaison members from other agencies, will not work when significant contributions are required from all participants. Another illustrative impediment is the bureaucratic tendency to retain the best talent for internally important jobs and assign the worst performers to interagency teams. Once the team has been formed, it must quickly establish workable policies and practices to replace those that a common organization and culture normally provide.

New team members people are unlikely to have ever worked together before, but now have an urgent need to collaborate quickly and effectively to deal with the crisis. Team members tacit understandings about how things get done in their "old" organizations no longer apply. The members need to trust one another to work effectively.

Because of the need to support and interact with humans in the team formation and collaboration process, agents represent a natural approach if they do not get in the way of the interaction. While we cannot hope to provide the smooth, voice-driven, intelligent assistant illustrated in the classic 1988 vision video, "Knowledge Navigator" by Apple Computer, that video certainly inspired us.

Our goal in this development was not to replace human judgment in forming teams with software agents. Agents will not, in the foreseeable future, be intelligent enough to be useful in such a role. Rather our goal was to leverage human intelligence in areas where it was needed and off-load the more routine or uninteresting tasks to software agents that don't mind dull jobs and perform them quickly and accurately.

This paper describes work that was undertaken to assist with the first problem area: setting up a cross-agency team efficiently. Our on-going work to assist in the second problem area and to improve our implementation in the first area will be reported on in the future.

2 Related Work

There have been many efforts to develop the basic notion of software agents facilitating the efforts of teams and the interests of the people in the teams [4]. The most notable is the work by Tambe and colleagues [12] on facilitating human collaboration and adjustable autonomy in teams. The resultant framework, Teamcore, assists in the functioning of teams consisting of both software and human agents, with the software agents imbued with the inherent ability to reason about their role in a team. There is no support for the formation of teams, however.

A similar comprehensive framework that can support teams is GPGP/TÆMS [8]. It provides domain-independent coordination for small groups of agents, and supports a variety of coordination strategies, including social laws. The result is optimal or satisficing goal-directed solutions to multiple concurrent problems. Unlike the teamwork facilitation we present herein, the focus is on software agents and not support for humans.

A related and complementary system to ours is the RETSINA multiagent infrastructure framework [3], which is based on four types of agents: (i) interface agents, (ii) task agents, (iii) information agents, and (iv) middle agents. Notably, the interface agents enable humans to be part of a system. Earlier experiments on a DARPA project proved that CoABS Grid agents could interoperate with RETSINA agents.

Nair *et al.* [10] pursue an alternative to ours for handling policies: they model policies as partially observable Markov decision processes, whereas we model them as defeasible rules in deontic logic.

In order to fulfill the goals and objectives of an organization, its members must understand their responsibilities and authorities, and must behave cooperatively. Liu and Dix [9] argue that the keys to the coordinated achievement of organizational goals are the norms that define the responsibilities and authorities for each human and establish regularities of behavior. They define the norms that software agents providing computer-supported cooperative work (CSCW) must facilitate and perform autonomously on the user's behalf. Our work provides an implementation and evaluation of their norm-based approach to CSCW, including their use of deontic operators to specify norms.

3 Analysis

3.1. Distributed Collaboration

To understand the requirements for a practical collaboration system, we developed a set of scenarios centered on the formation of an interagency crisis team. Each scenario assumed that a crisis had occurred somewhere in the world and that an appropriate decision maker had decided to form an interagency crisis team to deal with the problem. We modeled each agency as an independent entity with its own rules for determining what specific staff would serve on an interagency team. The rules for each agency were "hidden" behind an Organizational agent that provided the external interface to interact with that agency. This approach allows for great flexibility in dealing with real organizations.

In our scenarios, people continue to interact with one another by means of conventional tools, such as email. We augmented these conventional systems with software agents to assist in coordination and automate simple tasks. People use common office automation tools (email, browsers, and databases), software agents, and policies to interact and enable the rapid formation and approval of a team.

We recognize that people are not going to switch from the general purpose email, word processing, and other office automation tools they currently use. A collaboration system that attempts to replace the user's current desktop environment with a set of tools duplicating existing functionality will not succeed. Users will resist, and it is unlikely that such a system would ever win widespread acceptance on its merits. We believe that collaboration software must work in harmony with existing office automation software and the agency's supporting infrastructure.

Our agents manifest themselves largely through existing user interfaces and add value by supplying additional "smarts" behind the scenes. For example, a personal agent may monitor a user's email for traffic that it can process autonomously. A user can delegate some responsibility to that agent and set policy for when the agent can respond automatically and when it must get further approval from the user.

Once the team is formed and approved, a Groove collaborative workspace is launched for team members, it is populated with the appropriate background documents, and tasks are assigned to team members. We want agents to be able to notice when tasks are assigned to people and to monitor the status of those tasks in a way that is helpful to humans (who have the ultimate responsibility to deliver). The interactions between an agent and a human needed to accomplish this must be very simple. For example, suppose that five tasks have been identified in a virtual team meeting and the team leader wants to assign them and ensure that they are tracked. The team leader wants to be able to describe each task to the automated system with just a phrase, similar to what he would do if he were writing the task list on a whiteboard: Most project management systems require completion of what is essentially a half-page form for each task with dependency relationships to other tasks, resource requirements, earliest start dates and latest end dates, etc. This is far too much structure. Users must be able to leave many fields blank, change the entries later, or delete them entirely. Automated agents and tools that process this information must be able to deal with incompleteness and inconsistency of the information people provide and still constructively and intelligently assist the team in getting the tasks accomplished.

3.2. Federation Issues

Individuals making up the crisis team will be subject to policies from a variety of sources: their agency, federal and state laws, divisions and branches within the agency, and the team. If we include software agents, policy should involve personal preferences of the agents' owners. There is no one central source for policy.

Applicable policy will vary over individuals and over the software agents acting on their behalf.

Software systems as well as individuals must operate in a variety of domains. A domain can be characterized by the rules or policies that govern the behavior of entities in the domain. A technology domain might require particular software systems or network protocols. Administrative domains will have different organizational authorities set policies for them and control their behavior. Each domain might use a different security policy to determine who can access data and how it must be protected. Collaboration requires mechanisms that support the ability of our automated systems to cross all these domains. We must also support the ability of people to conform to the variety policies that govern their behavior

4 Design Alternatives

4.1. Ontologies as interfaces

A primary problem that we need deal with is the heterogeneity of the environment within which out system must function. In order for objects to interact at all we need to select at what level we would standardize interfaces in order for out objects to interact. Here we were faced with a large space of possibilities: message formats, RDFS, XML Schema, CORBA, RMI interfaces, FIPA messages, RDFS, CoABS, JAXB, and so on.

Rather than standardize on particular API's or object interfaces, we chose the approach of orienting our interface design primarily around of RDFS metadata. RDF has currency as the primary technology behind current efforts in the Semantic Web. It has the support of a research community and a set of emerging tools. RDF integrates well into the conceptual structure of the World Wide Web that has proven such a practical success.

We were also attracted by the simplicity of RDF's underlying mathematical model. The uniform representation of information as sets of triples of symbols [11] is remarkably elegant and maps naturally to many convenient interpretations. They can be readily interpreted as graphs, database tables, binary relations, expressions, object schema, and data. Their underlying uniformity and simplicity makes them a natural choice for automated reasoning systems that treat the tuples as well formed formula's is the sense of formal logic and infer new sets of the tuples from them. Perhaps because of it's minimalism, RDF seems to move between these different uses and concerns with remarkable ease.

Having selected RDF as our basis for development, we wanted to leverage existing ontologies expressed in RDF. Part of our previous effort had resulted in an RDF based version of the Rei policy language and we saw that we could leverage this technology for our own policy development. We also had available rule sets developed for XSB/Flora to automate reasoning about ontologies expressed in OWL, itself an extension of RDF with particular semantics associated with it. Thus we saw the technical means to easily obtain a fairly powerful reasoning engine to process policy statement written in Rei.

4.2. Task Management

We knew we wanted some form of task management to track subtasks assigned to individuals in the team and help see that things got done. However, typical workflow or project management systems are not appropriate to the task at hand. We believed that they required too much interaction from the user and assumed a long term view of the benefits of automating routine business processes. We believe that most workflow systems try to take too much ownership of workflow management and scheduling away from the human collaborators by querying them for too much information and making too many decisions for them.

Consider for example a workflow system that assigns task to individuals based on their skills, skill requirements of the tasks, task priorities, and the individual's current workload. Such a system needs an accurate model of what an individual's workload, a good encoding of skill requirements to tasks, and rather specific description of skills required for the task. Providing a system with all this information poses quite a burden. A fast moving inter-agency team cannot be assumed to have systems in place to support this information hungry process. We require a workflow system that can assist based on necessarily incomplete and lightweight human interactions and still provide valuable assistance in tracking workflow and making constructive suggestions to improve workflow within the team.

In keeping with our ontology-based approach, we developed an RDFS model of the concepts relevant to workflow. A team workflow agent is responsible for tracking workflow within the team. Agents interact with the workflow agent through messages representing their data as RDF triples.

4.3. Policy

Our first decisions with respect to policy dealt with how broadly we wanted to apply the term. We decided that policy can be divided into 3 types:

- Permissions -- access control or who can do what

- Configuration settings – typically specific to agents and applications

- Obligations – actions required by policy under certain conditions

We decided we would try to address all of these within a single conceptual framework, even if the mechanisms realizing them might differ.

Policies surrounding permissions (who can do what) are among the most studied. Some of the key concerns are where policy decisions get made (*policy decision point*) and where they are enforced (*policy enforcement point*).

Configuration settings are often not considered part of policy due to the difference in mechanisms delivering configuration settings and those that deliver access control information. We take a behaviorist view and consider such settings a part of policy because they have the same controlling effect on agents that other policies have.

Obligations are less traditional part of our policy framework. Agents may incur responsibilities as the result of accepting tasks from other agents through *speech acts* in Rei framework. Failure to carry out obligations may result in sanctions such a decrease in the degree of trust an agent has from certain parties to carry out certain actions.

4.4. Rei language

Rei is a declarative policy language for describing policies over domain actions [5]. It has an RDFS representation but includes a flavor of logic, in that it incoporates the notion of logic like variables for describing a wide range of constraints that may not be directly possible in existing ontology languages like RDFS, DAML+OIL or OWL. This provides greater flexibility in describing policies. An example of a Rei policy is, 'All entities in the same group as John have the right to perform a printing type of action on B/W printers in this lab'.

Rei is modeled on deontic concepts of rights, prohibitions, obligations and dispensations. We believe that most policies can be expressed as what an entity can/cannot do and what it should/should not do in terms of actions, services, conversations etc., making our language capable of describing a large variety of policies ranging from security policies to conversation and behavior policies. The policy language has some domain independent ontologies but will also require specific domain ontologies. The former includes concepts for permissions, obligations, actions, speech acts, operators etc. The latter is a set of ontologies, used by the entities in the system, which defines domain classes (person, file, deleteAFile, readBook) etc. and properties associated with the classes (age, num-pages, email). Though Rei itself has an RDFS representation, Rei allows domain specific information to be described in different ontology languages including RDFS, DAML+OIL and OWL, as it incorporates the required language reasoners as well.

The policy language supports individual policies as well as group and role based policies in a uniform manner by allowing domain dependent representations for roles and/or groups to be included in the conditions of the policy rules.

As the probability of conflicts in policies in distributed systems is high, Rei includes two constructs for specifying meta-policies that are invoked to resolve conflicts; setting the modality preference (negative over positive or vice versa) or stating the priority between policies. For example, it is possible to say that in case of conflict the Federal policy always overrides the State policy.

Associated with the policy language is a policy engine that interprets and reasons over the policies, related speech acts and domain information to make decisions about rights, prohibitions, obligations and dispensations of users. The engine is also capable of answering other queries related to policy making: who can perform a certain action, who can perform any action on a certain resource, what unfulfilled obligations does entity X currently have, what prohibitions do a certain class of users have etc.

5 Implementation

Our implementation uses both Java-based and C++ based agents and web services that interacted via the Jini-based CoABS Grid [7]. An innovative aspect of our agent-based system is a template-based design. The concept or technical vision for the template-based design is illustrated in Figure 1. Templates represent blackboard-like data structures that contain essential aspects of history or state information about individuals, organizations, and teams. Various agents and services interact with the templates to accomplish their goals (such as specializing the expertise required by a team to match an assigned situation or monitoring the status of *ad hoc* or standing tasks) and store current state in the templates. The templates evolve as things change as a result of changes in the environment or changes resulting from agent actions. Agents are used for team formation and evolution, policy management, resource utilization, and analysis of individual and team behaviors.

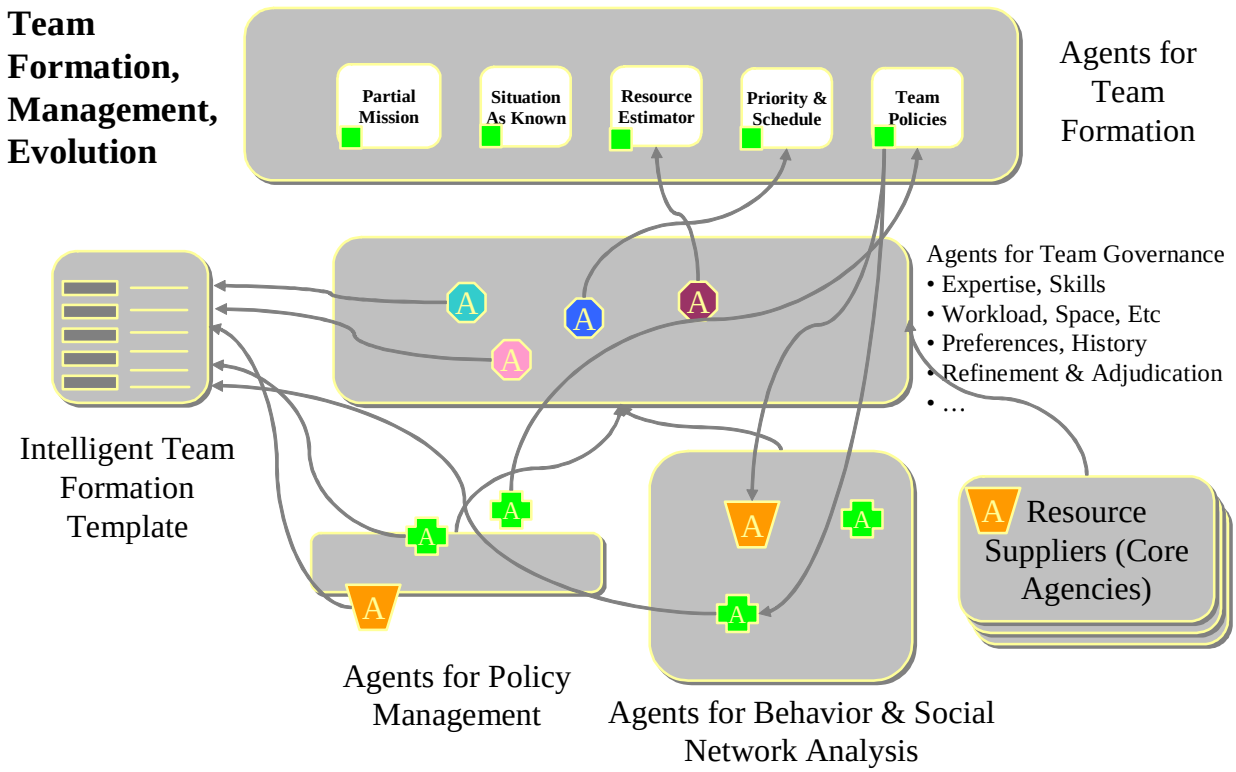

Figure 1: Technical Vision for Design

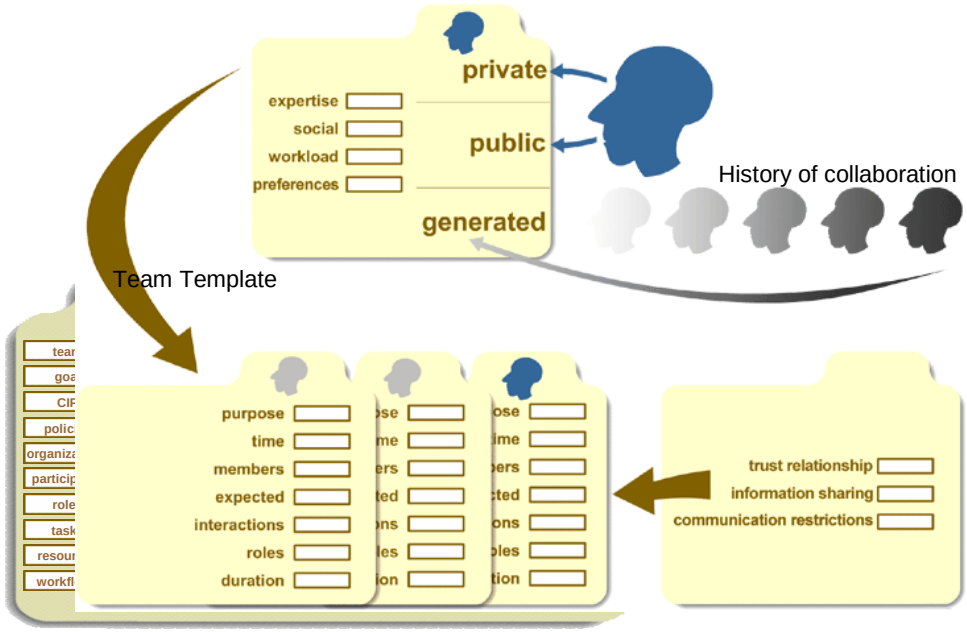

Figure 2 Templates Evolve via Interacting Agents

Figure 2 illustrates how our agents interact with and modify the templates representing the team, various organizations, and individual team member to accomplish their tasks. The data structures, shown as named slots in each template, represent the state of each attribute of the template. For example, the team participant's slot contains the names of all the current team members. These names are, in turn, linked to

templates describing each individual team member. Not all the information about each team member is available to the team, some is private and controlled by the individual, some is controlled by that individual's organization, and some is generated as a result of that individual's interaction with the team. Team policies, which are in part derived from the policies of each organization that is contributing staff to the team, will be discussed in more detail in the next section.

How we accomplish team formation is best illustrated by the graphical depiction of the agent interactions in Figure 3. The activities shown in Figure 3 are defined in Table 1 via their activity sequence numbers. These two diagrams summarize a great deal of technical information which we will not repeat here in the interests of brevity.

6 Role of Policy

6.1. Effects of Different Policies and Security

We want to ensure that the agents are responsive to policy changes and that security requirements are enforced. To this end, the system supports the correct robust behavior under the following conditions:

- Whether "report back before team finalization" is required
- Establishment of team policies that reflect de-confliction of different organizational policies.
- Different organizational evaluations of team effectiveness and policies regarding visibility into past problems.
- Different information sharing policies that constrain what can be shared between particular organizations and the team and restrict subsequent sharing outside of a team.

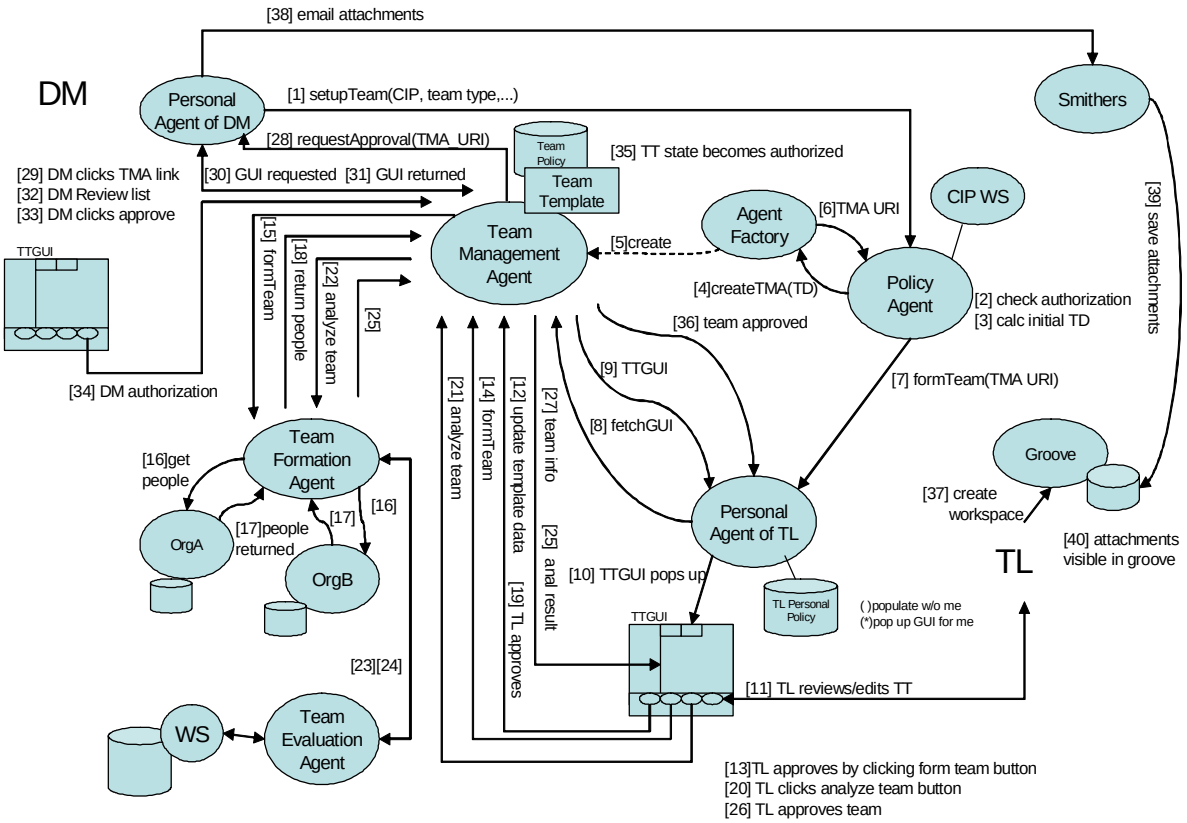

Figure 3 Agent Realization of Team Formation

Table 1 Team Formation Use Case and Activity Sequence

1	Decision Maker (DM) emails a message to his organization's Policy Agent to authorize Joe Team-Lead (TL) to set up and lead a cross-agency team that will respond specified crisis (ID 2741Q). He specifies that he wants to review the proposed team members before the team is finalized and that standard reporting policies should be required.
2	Policy Agent validates email signature and that sender authorized.
3	Policy Agent fills in the team template with initial data drawn from specified crisis information package (ID 2741Q)
4	Policy Agent instructs Agent Factory (AF) to create a Team Management Agent (TMA) using data contained in the initial team template
5	AF creates a new TMA with the specified data.
6	AF sends Policy Agent a pointer (URI) to the newly created TMA
7	Policy Agent forwards the original DM message along with a pointer (to locate the new TMA) to the designated TL. The Personal Agent (PA) routinely monitors email to the TL for messages that it recognizes so it detects the Policy Agent message and begins acting on it.
8	TL's Personal Agent sends a message to the TMA to launch a GUI on the TL' computer
9	TMA sends the GUI to TL's computer
10	GUI starts up on TL's workstation and displays the information in the initial team template
11	TL reviews team and makes changes via the GUI
12	GUI sends update message with new data to TMA
13	TL clicks "Form team" button on GUI
14	GUI sends "Form team" message to TMA referring to previous data
15	TMA sends "Form team" message to Team Formation Agent (TMA)
16	TFA sends messages to various Organization Agents that request each organization to provide candidates for selected positions on the team.
17	Organization Agents return lists of people to TFA
18	TFA combines lists and sends it to TMA for team template.
19	Changes to team template propagate out to the GUI for TL to see
20	TL selects "Analyze Team" button on the GUI
21	GUI sends "Analyze Team" message to the TMA
22	TMA receives message and delegates it to the TFA
23	TFA sends message to Team Effectiveness Agent for analysis of team makeup.
24	Results of analysis propagate beck to TFA, TMA and are written to team template
25	Team template changes propagate out to the GUI and are observed by the TL.
26	TL approves result by clicking button on GUI.
27	GUI transmits approval event to TMA
28	Team policy dictates DM approval, so TMA sends message with Team URI to DM's Personal Agent.
29	DM clicks on link in message to fetch GUI.
30	Request to Fetch GUI is transmitted to TMA
31	GUI is constructed by TMA and sent to DM's workstation.
32	DM reviews list of people on team via GUI
33	DM clicks Authorize button to approve.
34	Authorization transmitted to TMA
35	TMA changes state of team from proposed to authorized.
36	TMA sends email to TL indicating team approval
37	TL creates a Groove collaboration workspace for the team and his agent invites new team members to join that space.
38	DM sends team background documents via email to Smithers, the Group Mail Capture Agent (GMCA)
39	Smithers (GMCA) puts attachments into a directory visible to Groove
40	Documents are now visible to team members

- Unauthorized persons cannot form a cross-agency team
- Rogue agents cannot impersonate other agents, send false messages to another agent or replay messages.

More information on the policy aspects of the system is available in [2]

6.2. Agents and Policy

Before the process of forming a forming a new team is begun, the Team Management agent does not yet exist. Each team has a unique Team Management Agent. The Team Management Agent is created by an Agent Factory in response to request to form a new team. The request is made to a Policy Agent that acts as a gatekeeper to the (Team Management) Agent Factory and enforces organizational policy about who can form a crisis team, what an initial template for the a team will look like, and the initial team policy.

While there is a policy associated with the organization, and another associated with the team, there is also policy associated with the individual. That policy determines what capabilities or rights are delegated to that user's personal agent. In our system the personal agent can monitor the email of a user and respond automatically to correspondence on behalf of the use. This personal policy governs how much or how little

authority the individual grants to the personal agent. For example, we use it in the scenario to control whether the team lead's personal agent need ask for the team leader's approval on the list of roles (skills) required for a team, before acting to fill those roles.

The task of actually finding people to fill the roles making up the team falls on the Team Formation Agent. This agent is responsible for going outside of the organization to find people with the necessary skills. The Team Formation Agent is not allowed to look through the personnel records inside another organization. Instead the Team Formation Agent must negotiate with other Organization agents who will communicate to the Team Formation Agent what individuals those outside organizations has selected to for the team. Each organization may have its own policies and procedures for deciding who it wants to appoint to the team. These Organization Agents are the integration point for such federated decision making.

7 Initial Results and Assessment

7.1. Team Formation Speedup

Based upon our discussions with various individuals and managers associated with the traditional establishment of ad hoc cross-agency teams, team formations take at least hours and often weeks, depending upon the size, complexity, and expected duration of the team. In our demonstrations with human decision makers and team leaders, establishing a team took about twenty minutes. The time was measured from the time the decision was made to form a team (to deal with a particular hypothesized event) until ten team members from three agencies had been identified, approved, contacted and integrated into a downloadable collaboration environment that is populated with initial background documents and assignments. This enormous difference in the time required for team formation between the traditional approach and our agent-based approach was mainly the result of removing the human element in organizations from the decision making process about which members of an organization would be assigned to a given cross-agency team. Whether such a practice will ever be a realistic alternative in organizations is open to debate but the potential for improvement is vast.

7.2. Architecture Qualities

Is our implemented architecture the best choice? It is difficult to determine whether any architecture is a good one, let alone the best one, without extensive real-world deployment. However, significant thought has gone into issues of fitting into organizational structures (particularly with respect to policies and personnel decisions on team commitments) and of scalability. For example, organizations control policy within their domain and, within a domain, policy is usually controlled in a hierarchical fashion. However, any given individual or agent may be a member of multiple overlapping domains and thus subject to multiple policies. We have made the simplifying assumption that, within our agencies, policies at the individual level have been deconflicted (or at least the conflicts have been identified for human resolution). Thus, when a cross-agency team is formed, a set of Team Polices can be determined [2].

Similarly, the Policy Server could easily become a bottleneck if it was asked to make permission decisions for all individual agent actions. We have addressed this in several ways: through multiple policy engines and through the use of embedded platform security mechanisms for enforcement. This latter approach is the subject of further effort, because policy enforcement is a major task [13] and [14]. The CoABS Grid itself has been the subject of many scalability experiments and has performed exceptionally well [6].

In the current implementation, we are relying on the simple restart capabilities inherent in the CoABS grid for reliability [7]. More extensive reliability mechanisms can be implemented when needed.

7.3. Caveats

The current implementation has a number of possible limitations. Our approach to determining the requirements of a team for a particular purpose depends upon the availability of machine understandable semantic tags that describe the situation and a set of rules that allow general requirements to be specialized to the particular situation, albeit with human tuning. Whether either of these approaches will be available in real organizations is an open question. If team formation is to be automated, this semantic information and rules must be available. OWL presents an interesting opportunity for use in this area.

We have implemented one of many possible methods of selecting candidate team members from within an organization. Ours does a simple optimization across skills and availability and assumes that on-line sources of such information are available within an organization. This is certainly not the case in many, if not most, organizations. We do not view this as a serious problem, because the selection process within an organization is essentially isolated from the rest of the system because the interface is through the Organization Agent. Thus it would be easy to substitute a different selection process behind this agent with no changes to the rest of the system.

Similarly, we have implemented simple algorithms to determine team effectiveness and identify any past problems with candidate team members. These algorithms assume that requisite information is available online. Certainly, in most organizations, that is not currently the case. Additionally, there is little evidence in the literature that it is possible to predict how a group of individuals will function as a team. Furthermore, many organizations explicitly state that employees with the requisite skills or title should be equivalent and that personal likes/dislikes or personality conflicts have no place on the job. While this makes it easy for an organization to assign members to teams, it is not necessarily true. Offering up candidates for cross-agency teams from an organization's "turkey farm" is not unheard of. Again the system is modular enough to permit different approaches to this sticky area.

8 CONCLUSIONS AND NEXT STEPS

Our prototype agent-based system supports rapid formation of customized ad hoc cross-agency collaboration teams. Preliminary results indicate a reduction in team-formation time of two orders of magnitude.

Our next steps will focus on improving the effectiveness of human collaboration teams in a virtual environment by facilitating interactions with software agent teams that enable continuous assessment and restructuring of the virtual teams. We hope to demonstrate significant improvements in collaboration effectiveness based upon quantitative metrics such as scaling over number of people, organizations, types of organizations, time to achieve results, maintenance over extended periods of time, types of participants (human and computer), and dynamics of the participation.

9 ACKNOWLEDGMENTS

This work was partially funded by the Space and Naval Warfare (SPAWAR) Systems Center (SSC) under contract N66001-03-C-8001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the SPAWAR System Center or the U.S. Government.

10 REFERENCES

[1] Shreedhar M., and G. Varghese, "Efficient fair queuing using deficit round robin," in SIGCOMM: ACM Special Interest Group on Data Communication, (Cambridge, MA), pp. 231–242, September 1995.

[2] Cornwell, M., Just, J., Kagal, L., Finin, T. A Policy Based Collaboration Infrastructure, submitted to IEEE 5th International Workshop on Policies for Distributed Systems and Networks, New York, June 7-9, 2004.

[3] Giampapa, J.A. and Sycara, K.. Team-Oriented Agent Coordination in the RETSINA Multi-Agent System. Technical Report CMU-RI-TR-02-34, Robotics Institute, Carnegie Mellon University, December 2002.

[4] Huhns, M.N. Agent Teams: Building and Implementing Software. IEEE Internet Computing, vol. 4, no. 1, pp. 91-93, January/February 2000.

[5] Kagal, Lalana et al. A Policy Language for A Pervasive Computing Environment, IEEE 4th International Workshop on Policies for Distributed Systems and Networks, June 2003.

[6] Kahn, M., et al, "CoABS Grid Scalability Experiments", J. Autonomous Agents and Multi-Agent Systems, 7, 171-178, 2003, Kluwer Academic Publishers, Netherlands.

[7] Kahn, M., et. al., DARPA CoABS Grid Users Manual V3.2.1, October 2001. http: coabs.globalinfotek.com

[8] Lesser, V. et al., Evolution of the GPGP/TAEMS Domain-Independent Coordination Framework. Proc. AAMAS, 2004.

[9] Liu, K. and Dix, A. Norm governed agents for CSCW, 1st Int'l WS on Computational Semiotics, Paris, 1997.

[10] Nair, R., et al. Taming Decentralized POMDPs: Towards efficient policy computation for multiagent settings. Proc. IJCAI, 2003

[11] Notation 3. http://www.w3.org/DesignIssues/Notation3, 2001.

[12] Pynadath, D. and Tambe, M. Automated teamwork among heterogeneous software agents and humans. J. Autonomous Agents and Multi-Agent Systems. 7:71-100, 2003.

[13] Suri, N, et al. DAML-based Policy Enforcement for Semantic Data Transformation and Filtering in Multi-Agent Systems. Proc. AAMAS, July 2003.

[14] Uszok A., et al. KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction, and Enforcement, IEEE 4th International Workshop on Policies for Distributed Systems and Networks, June 2003