Coalition Agents Experiment: Multiagent Cooperation in International Coalitions

David N. Allsopp, Patrick Beautement, and Michael Kirton, *QinetiQ*
Jeffrey M. Bradshaw and Niranjan Suri, *University of West Florida*
Edmund H. Durfee, *University of Michigan*
Craig A. Knoblock, *University of Southern California*
Austin Tate, *University of Edinburgh*
Craig W. Thompson, *Object Services and Consulting*

The Coalition Agents Experiment aims to show that multiagent systems offer effective tools for dealing with complex real-world problems by enabling agile and robust coalition operations and interoperability between heterogeneous military systems.

Military success requires executing high-tempo, coherent, decisive actions faster than an opponent can react—that is, decision dominance through command agility. Command agility means having the flexibility to grasp fleeting opportunities and being innovative, creative, and unpredictable in a manner that (even if low-tempo) increases confusion in the opponent's mind. This process is command-led, with human decision making primary and technology playing a secondary role. Shared understanding and information superiority are key enablers in this process and are fundamental to network-centric warfare (www.dodccrp.org). In addressing interoperability requirements, we must also address data security, control over semitrusted software from other coalition partners, and the resulting system's robustness—for example, its ability to withstand denial-of-service attacks.

However, good decisions depend on good data, which can become a problem as mission complexity increases. Military coalitions—large-scale, multifaceted, multinational, virtual organizations—often must be rapidly created and changed as circumstances shift. In addition to integrating single-service and joint capabilities into a coherent force, coalition operations must rapidly configure foreign or legacy systems into a cohesive whole. Yet current coalition operations often suffer from data overload, information starvation, labor-intensive information collection and coordination, and stand-alone stovepipe command systems that use incompatible data formats. This leads to a horrendous technical integration task and offers commanders only scattered snapshots of the battlespace.

In the inevitable absence of preexisting coordinated systems, we must take a rapid, flexible, on-the-fly approach that permits assembling capabilities at runtime. We believe agent-based computing (described in the sidebar, "Software Agent Technology") offers a promising new approach to effective coalition operations because it embraces the coalition environment's open, heterogeneous, diverse, dispersed nature. This article describes how software agents acting on behalf of human users enable military commanders to act decisively in cyberspace and thus contribute to "cyberspace superiority," a critical component of warfare in the information age.[1] We focus here on the rapid integration of agents and legacy systems to improve interoperability and support human situational awareness and decision making, rather than, for example, sophisticated teamwork and planning between agents.[2]

CoAX project goals

The Coalition Agents Experiment (CoAX; www.aiai.ed.ac.uk/project/coax) is an international collaboration carried out under the auspices of DARPA's Control of Agent-Based Systems program (http://coabs.globalinfotek.com). Building on the CoABS Grid framework, the CoAX agent infrastructure groups agents into domains reflecting real-world orga-

1094-7167/02/$17.00 © 2002 IEEE

nizational, functional, and national boundaries such that security and access to agents and information can be governed by policies at multiple levels. CoAX, begun in February 2000, seeks to demonstrate that the agent-based computing paradigm offers a promising new approach to the technical issues of establishing coherent command and control (C2) in a coalition organization. Research hypotheses include the following:

- Agents offer a useful metaphor for dealing with the complexity of real-world systems such as military operations.
- An agent-based C2 framework can support agile, robust coalition operations.
- Software agents can enable interoperability between legacy or incompatible systems.
- The CoABS Grid can quickly integrate many different agents and systems, permitting rapid creation of virtual organizations.
- Domain policies can structure agent relationships and enforce coalition policies.
- Intelligent task and process management can improve agent collaboration.
- Semantic Web technology can improve agent interoperability between disparate coalition command systems.

The CoAX team has built a software agent testbed based on the CoABS Grid. This article describes the work done, the demonstrations carried out so far, the scenario and storyboard used, and some insights gained.

A representative scenario and coalition command structure

The CoAX team is conducting a series of demonstrations of increasing complexity in a stylized yet realistic peace-enforcement scenario situated in Binni, a fictitious African state. These demonstrations use agent technologies to build a coalition command system for intelligence gathering; visualization; and campaign, battle, and mission planning and execution.

The scenario

To create a suitably realistic scenario for the experiments, we expanded the fictional Binni scenario developed for The Technical Cooperation Program (www.dtic.mil/ttcp).[3] In this scenario, set in 2012 on what is currently the Sudanese Plain, global warming has altered the world's political balance. As previously uninhabited land has become arable, the area has received much foreign investment and is now called "the Golden Bowl of Africa."

Software Agent Technology

Agents can be viewed as semiautonomous software designed to help people cope with the complexities of working collaboratively in a distributed information environment. A community of agents works as a set of distributed, asynchronous processes communicating and sharing information by message passing in some infrastructure. Essentially, agents communicate with users and among themselves to find, format, filter, and share information, and they work with users to make the information available wherever and whenever they need it. Agents can also suggest courses of action proactively, monitor mission progress, and recommend plan adjustments as circumstances unfold. They provide the modularity and abstraction required to tackle large and complex problems.[1]

Because the agent paradigm offers a good way of building complex software systems in general, it offers potential benefits in the coalition setting.[2] To this end, DARPA's CoABS program has created the CoABS Grid, a middleware layer based on Java and Jini technology that provides the computing infrastructure to integrate heterogeneous agent communities and systems rapidly (http://coabs.globalinfotek.com).

References

1. K. Sycara, "Multiagent Systems," *AI Magazine,* vol. 19, no. 2, Summer 1998, pp. 79–92.
2. N.R. Jennings, "An Agent-Based Approach for Building Complex Software Systems," *Comm. ACM*, vol. 44, no. 4, Apr. 2001, pp. 35–41.

A conflict has developed between two countries in the area. Gao, to the north, has expansionist aspirations but is only moderately developed, possessing old equipment and a mostly agrarian society. Agadez, to the south, is a relatively well developed, fundamentalist country. Gao has managed to annex an area of land, name it Binni, and establish its own puppet government, which has come under fierce attack from Agadez. Gao, playing the "threat of weapons of mass destruction from Agadez" card, has enlisted UN support to stabilize the region (see Figure 1). We adapted this basic scenario for several CoAX demonstrations, beginning with the initial planning phase and moving to shorter timescales and more dynamic, uncertain events for the execution phase.

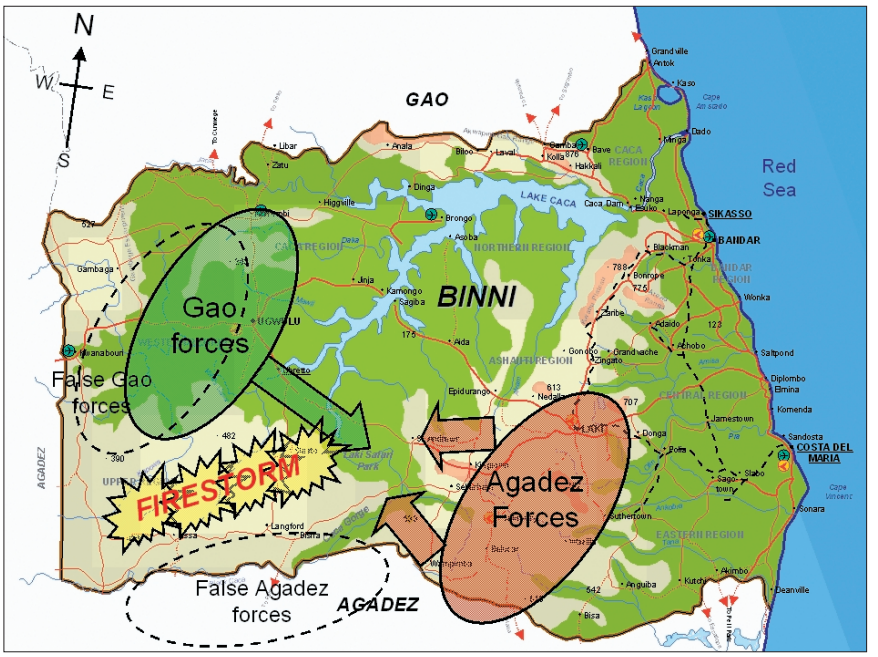


Figure 1. Map of Binni showing firestorm deception. Misinformation from Gao is intended to displace the firestorm to the west, allowing Gao and Agadez forces to clash in the region of the Laki Safari Park.

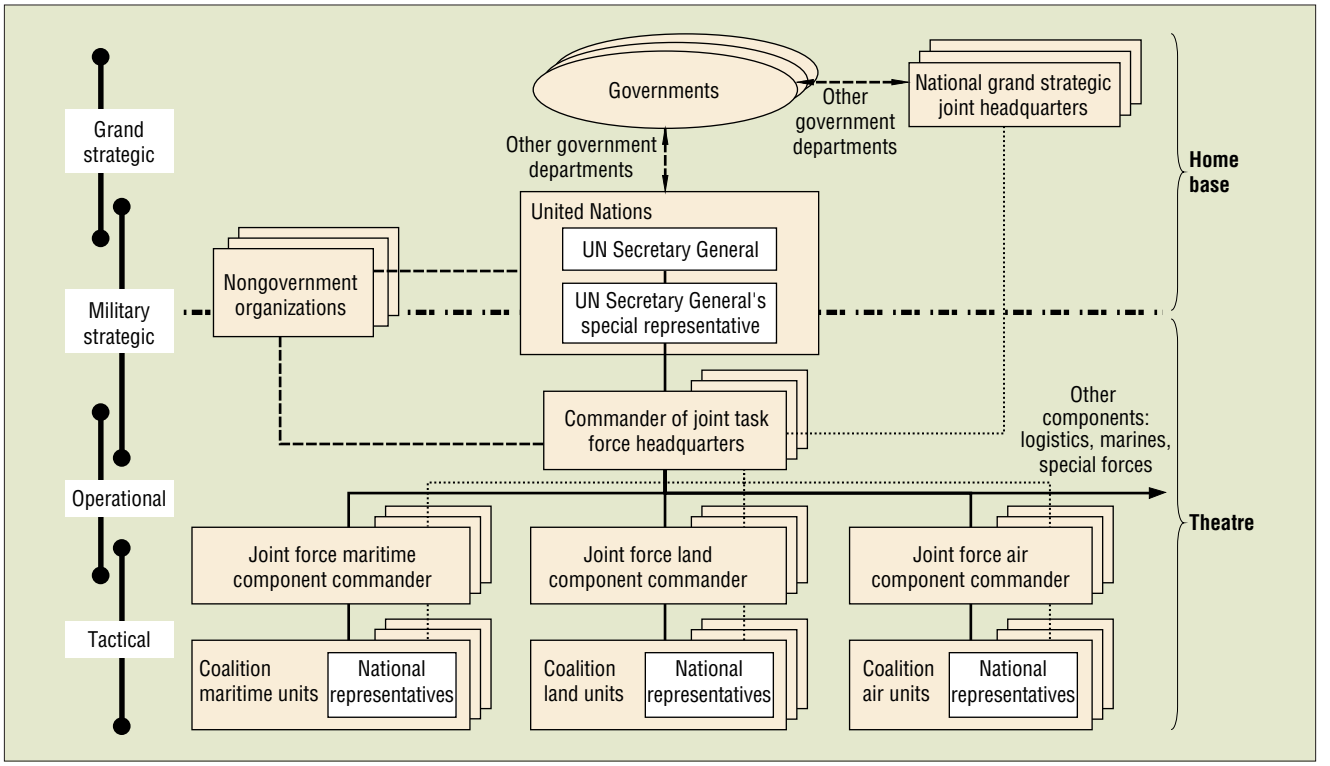


Figure 2. This representative coalition structure shows the chain of command down from the United Nations. The solid black lines show the legal lines of authority (the command chain) and accountability. Dashed lines show an advisory or negotiating role; dotted lines show the "ghosted" command chains of the participating nations. The approximate command level at which the various entities operate appears on the left.

Coalition command structure

In this scenario, runtime composability offers a close metaphor for the coalition operations' dynamic uncertainty and therefore should suitably test the software agents. This coalition operation needs to rapidly configure various incompatible systems into a cohesive whole within an open, heterogeneous, dispersed environment. We can best illustrate the situation's complexity through the Binni coalition command structure, shown in Figure 2.

This representative and realistic coalition command structure involves the UN, governments, other government departments (such as the Foreign Office), nongovernment organizations (such as Oxfam), representatives of all the coalition countries (with their own "ghosted" command structures, shown as dotted lines), and the coalition headquarters and subordinate fighting forces. The participants would likely agree to such a coalition structure: no specific country owns any part of the formal command chain, and "levels of command" overlap with no rigidly defined boundaries. Dashed lines show an advisory or negotiating role.

Software agent architecture

Integrating information across a coalition involves more than just deploying technology—it also involves creating a coherent "interoperability of the mind" at the human level, where many social and cultural factors come into play. We thus find no straightforward mapping between the human and technical worlds.

Human domains

From the human perspective, we identified four types of domains:

- Organizational domains, such the joint task force headquarters (JTF HQ)
- Country domains, with each national command chain a separate, self-contained domain
- Functional domains, where entities collaborate on common tasks such as meteorology or intelligence
- Individual human domains of responsibility, where commanders have responsibility for their own HQ and all subordinate ones (in practice, they delegate); hence these domains might overlap

These domain types are not entirely exclusive, and they overlap and interact at many different levels depending on the viewpoint taken. This complexity at the human level creates difficulties for technical systems.

Software agent domains

At the most basic level, the agents and systems to be integrated require infrastructure for discovery of other agents and messaging between agents.

CoABS Grid infrastructure. The CoABS Grid provides this infrastructure. Based on Sun's Jini services, which are based on Java's Remote Method Invocation, the Grid allows registration and advertisement of agent capabilities, and communication by message passing. Operators and even the agents themselves can add or remove agents on the Grid or update their advertisements without network reconfiguration. Agents that fail are automatically purged from the registry. The system can use multiple look-up services, located by multicast or unicast protocols. In addition, the Grid provides logging, visualization, and, more recently, message encryption and agent authentication.

The system advertises and matches agents using one or more Jini lookup services (LUSs), which are configured and run using a simple GUI tool that also controls the required HTTP class server and the Remote Method Invocation daemon. Agents can discover LUSs using a multicast protocol over a local area network or unicast outside the local network; they are

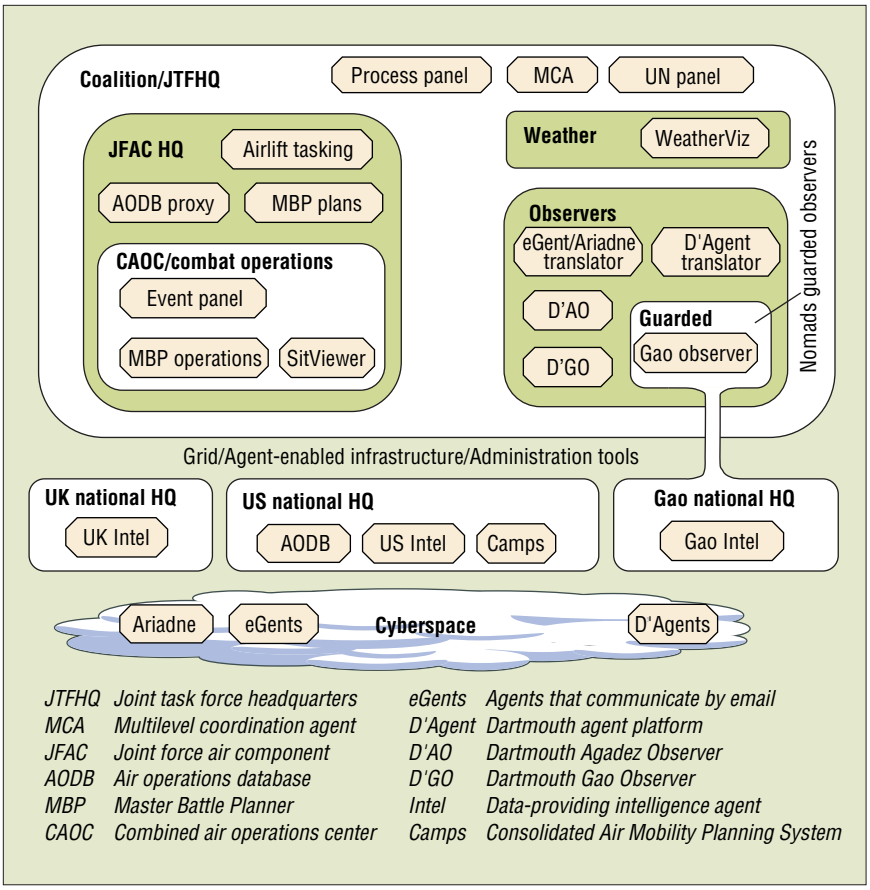


Figure 3. Typical CoAX domain structure. Rounded rectangles indicate domains; angled rectangles, agents. Some agents act as proxies for other agents or legacy systems that are not themselves domain-aware. Each domain would also contain a domain manager agent and a matchmaker agent (omitted for clarity). Domain nesting indicates a hierarchy of responsibility and policy control.

assigned unique IDs when they register with a LUS. The LUS supports agent lookup using template-based matching (exact or wildcard) and predicate-based matching, and also allows researchers to substitute their own lookup algorithm. The Grid provides helper objects that hide Jini's complexity and provide access to lower-level objects as required. Grid helper classes queue messages between agents, which use the FIPA agent communication language (www.fipa.org). An agent can access messages by polling the queue or can choose to receive callbacks when messages arrive.

KAoS domain management. The increased intelligence that software agents provide is both a boon and a danger. Because they operate independently without constant human supervision, agents can perform tasks that would be impractical or impossible using traditional software applications. However, this autonomy, if unchecked, could also severely damage military operations if buggy or malicious agents arose.

The Knowledgeable Agent-Oriented System (KAoS) provides services to assure that agents from different developers and running on diverse platforms will always operate within the bounds of established policies and will be continually responsive to human control to permit safe deployment in operational settings.[4,5] KAoS services and tools permit policy specification, management, conflict resolution, and enforcement within the specific contexts established by complex military organizational structures.

KAoS domain management services can group agents into logical domains corresponding to organizational structures, administrative groups, and task-oriented teams. Within CoAX, these domains mirror the human domains described earlier, allowing for complex hierarchical, heterarchical, and overlapping structures. An agent domain consists of a unique domain manager instance along with any agents registered to it. Alternatively, an intentionally defined domain consists of a set of agents sharing one or more common properties (for example, all agents physically residing on some host). The domain manager manages agent registration and serves as a single point of administration and enforcement for domainwide, host-wide, Virtual Machine-wide, VM-container-wide, or agent-specific policies.

Domain policies. A policy is a declarative constraint governing the behavior of one or more agents, even those that might not be domain-aware or might be buggy or malicious. For example, an operator can create a policy that all messages exchanged among agents in the joint force air component (JFAC) HQ domain must be encrypted, or that an agent cannot simultaneously belong to the US and the UK domains. A policy does not tell the agent how to perform its task, but simply specifies the conditions under which it can perform certain actions.

KAoS domain management includes policies governing authorization, encryption, access control, and resource control. However, our focus on agent systems requires implementing more advanced measures. For example, KAoS pioneered the concept of agent conversation policies,[4] where teams of agents can be formed, maintained, and disbanded through agent-to-agent communication using an appropriate semantics. In addition to conversation policies, we are developing representations and enforcement mechanisms for mobility policies, domain registration policies, and various forms of obligation policies. We represent these policies in ontologies using the DARPA Agent Markup Language, and we use an efficient description-logic-based approach as the basis for much of the domain manager's reasoning to discover and resolve policy conflicts and perform other kinds of policy analysis.

Separating policy specification from policy enforcement mechanisms allows policies to be dynamically reconfigurable and relatively more flexible, fine-grained, and extensible. Agent developers can build applications whose policies can change without necessarily requiring source code changes. Using declarative policies to describe and govern agent system behavior results in easier recognition of nonnormative behavior, policy reuse, operational efficiency, reactivity to changing conditions, and possibility of offline verification.

CoAX software agent domains

The CoAX demonstrations contain software agents grouped into agent domains using the CoABS Grid, with KAoS domain management services enforcing the policies. Figure 3 shows a typical domain configuration.

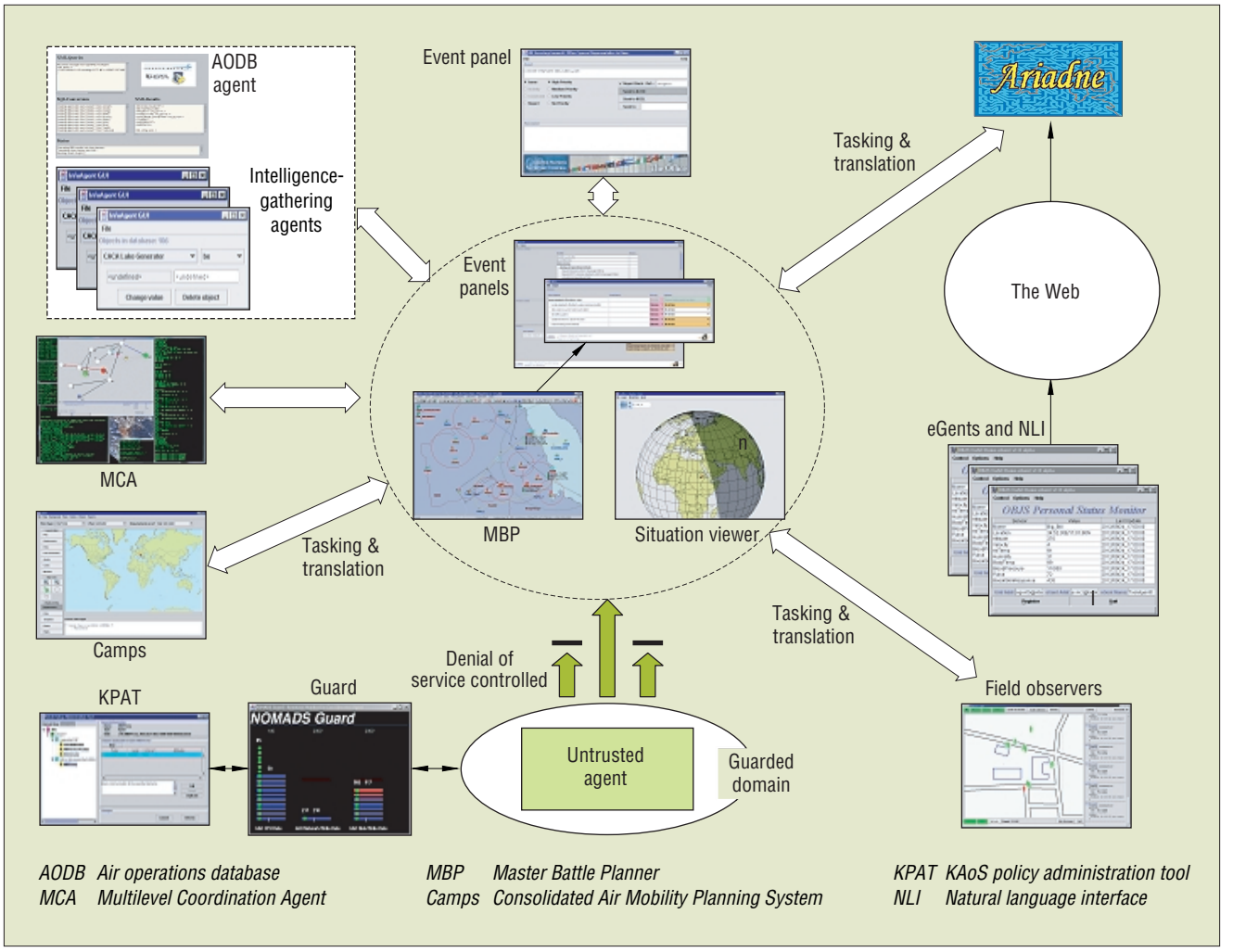


Figure 4. Overview of technologies and agents. The central visualization and planning tools find and acquire data (for example, disposition of ground forces) and services (for example, airlift scheduling and plan deconfliction, that is, detection and resolution of conflicts between two or more plans) from the other agents and systems, in some cases via intermediate tasking and translation agents.

Demonstration storyboard and technologies

As we progress through the Binni scenario storyboard (noted in italics), we'll describe each agent system and the technologies used in each scene. Figure 4 shows agent interactions from the technical point of view.

Domain population

Following the outbreak of hostilities, the UN has deployed forces to stabilize the region. Active coalition participants at this time include the UK, US, and Gao.

Agent domains set up using the CoABS Grid infrastructure and KAoS domain management services represent the organizational structures (the JTF HQ and JFAC HQ), the nations (UK, US, Gao), and various functional domains such as Weather and Observers. Numerous agents populate these domains, registering with their domain manager and optionally advertising their services with their domain matchmaker.

Data gathering and air planning

After exploring options to separate the opposing forces and restore peace in the region, the coalition has rejected the deployment of a large ground observation and peace enforcement force and other courses of action, and has chosen a "firestorm" mission. This will clear land to enable simpler remote and ground observations with less risk to the coalition peacekeepers. The coalition undertakes initial information gathering and planning.

Master battle planner. We perform air planning at the JFAC using QinetiQ's MBP (Master Battle Planner), a visual planning tool for air operations. MBP provides planners with an intuitive visualization on which they can manipulate air intelligence information, assets, targets, and missions using a map-based GUI (see Figure 5). This lets an operator build a battle scenario containing targets, offensive and defensive units, airspace measures, and other objects using simple dialogs and point-and-click techniques on the map. The operator can then move objects around on the map, change their properties, and display information such as unit allegiance, status, and ranges.

The operator can interact with these entities and plan individual air missions (or more complex mission packages) by dragging and dropping offensive units onto targets on the map. Supporting or defensive elements are added in the same way. The system gives the operator analytical tools to assess the planned air operations for

- The best use of resources (for example, by highlighting overtasked air units)
- Time-phasing (through charts and animated

"fly-out" that simulates the planned missions, showing where aircraft are expected to be at a given moment in the future)
- Concordance with military guidance given

We have agent-enabled MBP, a monolithic C++ application, by wrapping it in Java code using the Java Native Interface. This allows MBP to receive all scenario data (such as targets, assets, and airspaces) from multiple information-providing agents and update this information in near-real time. This process is integrated into normal MBP usage: when an operator views an object's status, agents are automatically tasked to update the information. Agents can also push status changes to MBP. MBP can accept and merge information on other air missions with those planned within MBP, as described later, and can save and export missions to let other agents reason with the data.

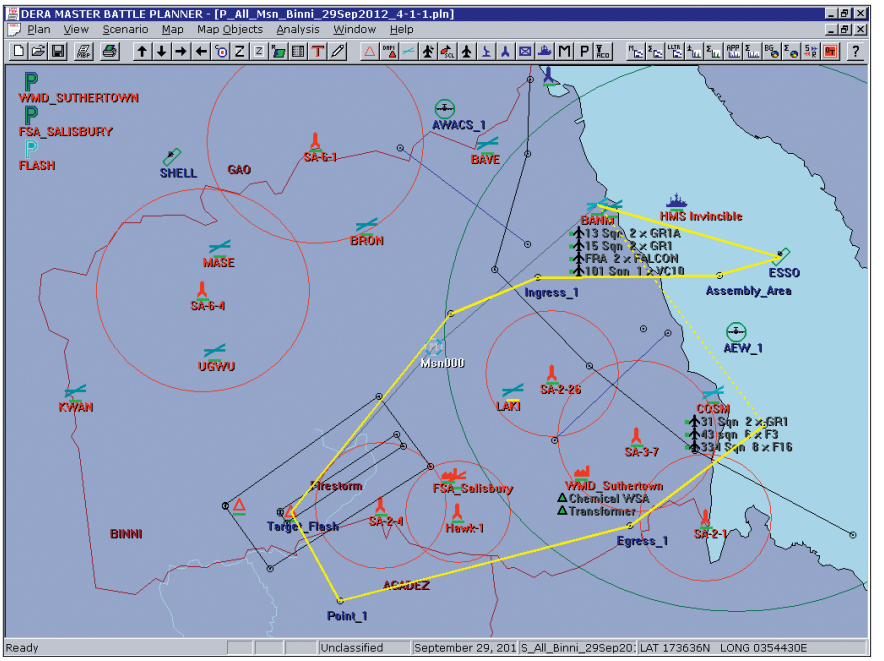


Figure 5. Master Battle Planner map display of Binni, Gao, and Agadez. A selected mission is highlighted, proceeding from an airbase (BANM) to refueling tanker (ESSO) to the target via waypoints and airspaces, and back to base by a different route.

Consolidated air mobility planning system (Camps). The second real military system integrated into the demonstration is the Air Force Research Laboratory's Camps mission planner, shown in Figure 6. Camps develops schedules for aircraft to pick up and deliver cargo within specified time windows. It takes into account aircraft capability constraints, port handling capabilities, crew availability and work schedule, and so on. Users develop plans (schedules) for aircraft to carry a particular cargo, specifying the intermediate ports, air refueling tracks, and the kinds of crews that will be available. They can also specify constraints on the airports potentially involved in plans under development.[6,7]

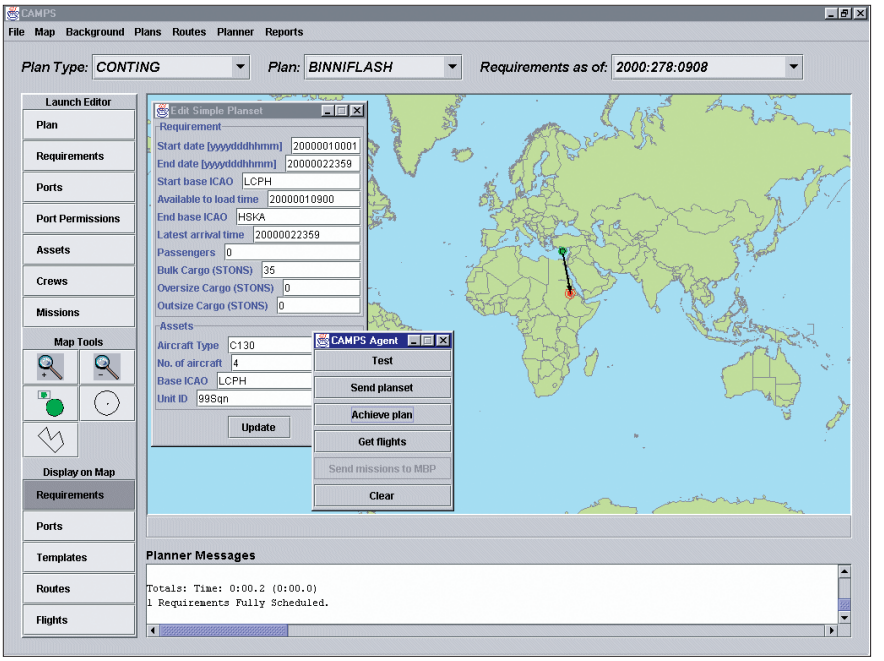


Figure 6. The Camps airlift planner, labeled "CAMPS," and the demonstration agent, including "CAMPS Agent" and "Edit Simple Planset," used to task the Camps agent with a simple requirement: Move cargo from Cyprus into Binni.

In the demonstration scenario, Camps schedules cargo airlifts into Binni. Because these airlift flights could conflict with offensive air missions, an intermediate agent requests scheduled flights from the Camps agent and sends them to MBP, forming part of the normal MBP air visualization. This intermediate agent tasks Camps and also translates between the Camps messages in KQML (Knowledge Query and Manipulation Language) and the MBP messages in XML.

In this example, only partial translation is possible because Camps and MBP differ fundamentally in their concept of air missions. A Camps mission consists of an arbitrary collection of flights, where *flight* means a single aircraft taking a single journey from A to B. An MBP mission, however, consists of a starting point and a route that must return to the starting point (perhaps by a different path) and might consist of multiple aircraft. Camps can therefore produce routes that have no fully valid representation in MBP, although they could be partially represented or indicated graphically.

Ariadne. In a similar manner, agents translate and forward to MBP Web-based weather information extracted by the University of Southern California's Ariadne system, again forming part of the normal air situation visualization. Ariadne facilitates access to Web-based data sources via a wrapper–mediator

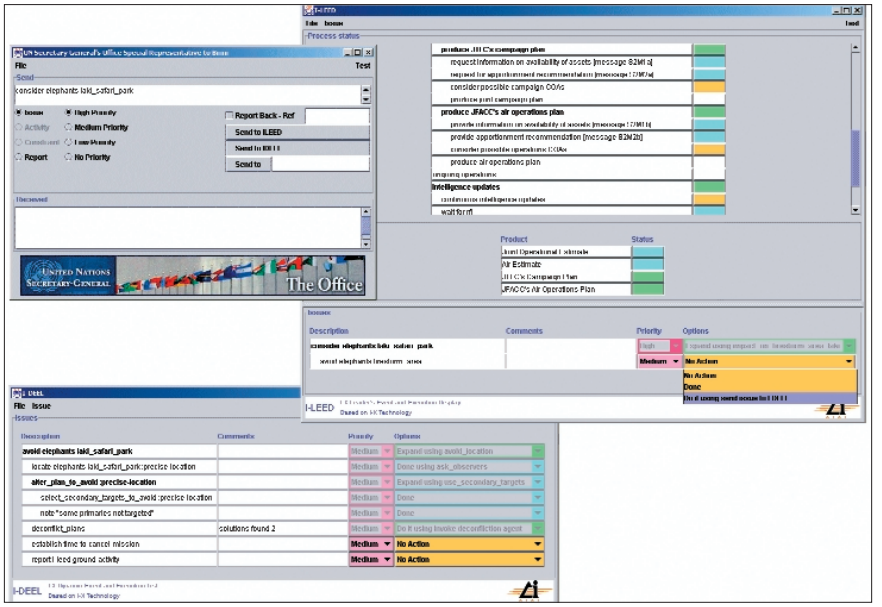


Figure 7. I-X process and event panels.

approach,[8] permitting rapid creation of wrappers that make Web sources look like databases. These interpret a request (expressed in SQL or some other structured language) against a Web source and return a structured reply. The mediator software answers queries efficiently using these sources as if they formed a single database. Custom code once handled translation of the Ariadne XML into that expected by MBP, but XSLT (Extensible Stylesheet Language Transformations) now handles this more easily, as we will show later.

I-X process panels (I-P^2). I-X process panels support this coalition planning process. The I-X research program seeks to create a well-founded approach to human–computer cooperation in the creation or modification of products such as a plan, design, or physical entity. In other words, it supports synthesis tasks, as well as more general collaborative activity. The I-X research draws on earlier work on O-Plan,[9] I-N-OVA,[10] and the Enterprise Project,[11] but seeks to make the framework generic, clarify terminology, simplify the approach taken, and increase core ideas' reusability and applicability. CoAX is using the I-X approach to provide task and process support and event-response capabilities to various coalition participants (see Figure 7).

An I-X process panel (I-P^2) acts as a workflow and messaging "catch-all" for the user. It can act in conjunction with other panels for other users if desired. A panel can take any requirement to handle an issue, perform an activity, or add a constraint (in the future). The panel deals with these via

- Manual (user) activity
- Internal capabilities
- External capabilities (invoke or query)
- Rerouting or delegating to other panels or agents (pass)
- Planning and executing a composite of these capabilities (expand)

It can also cope with partial knowledge. It receives reports and interprets them to understand the current status of issues, activities, and constraints; to understand the current world state, especially the status of process products; and to help users control the situation.

Resource control via domain policies. *Gao has host nation status within the coalition, but its intentions are unclear and it is distrusted. Coalition staff take special care to monitor the information passed to and from Gao within the coalition. During the demonstration, misinformation feeds by Gao (intended to displace the firestorm to allow Gao to take an advantage and move forward) are detected and thwarted. Gao becomes belligerent and launches a denial-of-service attack against the coalition's C3I infrastructure.*

The Gao agent in the demonstration runs under Nomads, a mobile agent system from the Institute for Human and Machine Cognition. The Nomads project aims to develop a set of distributed agent-based systems using the Java language and environment. The agent code runs in a new Java Virtual Machine, the AromaVM, which provides two key enhancements over standard Java VMs: capturing threads' execution state and controlling resources consumed by threads. By capturing threads' execution state, the Nomads system provides strong (transparent) agent mobility.

In addition, the resource control mechanisms let users control and allocate agents' resources and protect against malicious agents' denial-of-service attacks. When the Gao agent exceeds certain resource limits, a domain guard triggers an automatic domain policy change and instructs the AromaVM to reduce the resources available to the malicious agent (see Figure 8). An operator can manually reduce the limits further using the KAoS policy administration tool.

Data feeds from mobile devices and observers. *The coalition has planned the firestorm mission and aircraft have already taken off. However, the news media break a story that wildlife in an important safari park in Binni might be in danger because the park overlaps the firestorm area. With only an hour to go, the UN Secretary General's special representative to Binni asks the Joint Task Force commander to consider the wildlife risk aspects of the planned approach. Dynamic information gathering and information feeds using agent technology are employed to create a real-time feed of the position of some at-risk large mammals.*

Using the event panels, operators can note this urgent issue and break it down into subtasks. They can monitor aircraft progress in near-real time on the QinetiQ Situation Viewer and determine from MBP the time left before aircraft are committed to their targets. Data on animals' locations in the safari park is available online via agents running on monitoring devices attached to the park's large mammals. Operators can query historical data from these devices, called eGents (agents that communicate by email, developed by Object Services and Consulting), using a natural language interface. To aid the planners, Ariadne extracts data from the pages and a translator agent uses XSLT to create a live data feed from the safari park Web site. The resulting message stream, sent to MBP and to the Situation Viewer agent, allows visualization of the animals' current position and track (see Figure 9).

Another instance of the translator agent transforms ground force movement data from Dartmouth College's D'Agents field observation system. Visualized in the same way as the animal data, this data helps the coalition identify a convergence of hostile forces on the Laki safari park area.

Plan export and deconfliction. *After consideration, the commander decides to continue with the firestorm mission but to replan as necessary to avoid risk to wildlife. Air planners adjust firestorm targets or select secondary targets as necessary for the first wave of firestorm bombing. The system automatically detects the effects of these changes on the coalition's medical and humanitarian operations and avoids unintended conflicts between disjoint coalition operations.*

After revising the air plans using MBP, a translator agent then sends them to a deconfliction agent to check them against planned activities in other coalition HQs. The University of Michigan's Multilevel Coordination Agent processes the plans using multiple levels of abstraction to generate solutions.[12] I-X event panels keep planners informed of progress, and planners can view the results on the MCA display when ready (see Figure 10). Operators adjust the plans iteratively until they've resolved the conflicts.

Dynamic forced migration (scram) of observer agents. *Agadez seeks to use this complication to seize the initiative and launches fighter attacks against a valuable airborne system monitoring the operation. When coalition sensors detect this attack, the airborne system starts to regress. This means that the system's observer agents will not be able to continue providing information to the coalition.*

To solve this problem, the administrator uses the Nomads mobile agent system's forced migration (scram) capabilities to move the observer agents from the airborne system to a secondary ground station platform. Nomads uses the AromaVM state capture mechanisms to capture the observer agents' full execution state and then sends it to a new platform, where the agents can be restarted without any loss of their ongoing computations. This lets the observation agents continue operating from the ground station and provide information to the coalition even after the airborne system regresses.

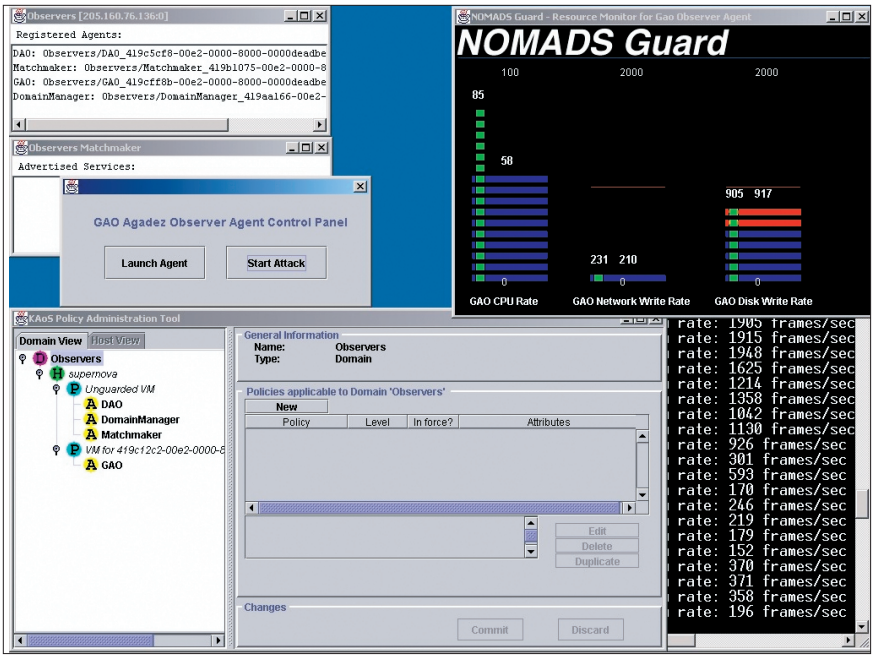


Figure 8. A denial-of-service attack by the Gao agent is starving other agents of resources (note the decreasing processing rate in the console, bottom right). The guard (top right) monitors the Gao agent's resource usage. The excessive resource usage triggers a domain policy change, which lowers resource limits enforced by the AromaVM. The policy can also be changed manually using the KAoS Policy Administration Tool (bottom left).

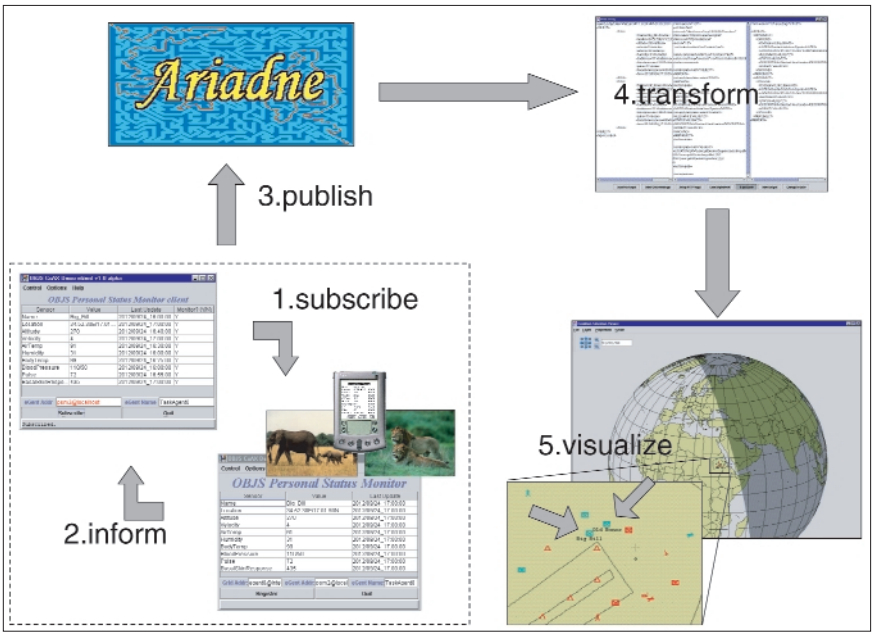


Figure 9. An eGent client subscribes to eGents running on mobile devices (wildlife tags). The client publishes data from these devices on a Web page. Ariadne extracts data from these pages and produces XML. The XML is transformed into another format by another agent using XSLT and finally is sent to agents such as MBP and Situation Viewer for visualization.

Software agent assessment

CoAX aims to show that the agent-based computing paradigm provides the computational support needed in coalition operations. The evidence so far confirms this view: our demonstrations show disparate agent systems working together in a realistic coalition application, with agents usefully sharing and managing access to information across a stylized coalition architecture. The CoABS Grid and KAoS domain management capabilities enabled us to interoperate, for the first time, previously stand-alone US and UK military systems as well as diverse agent-based information resources. In particular, the CoABS Grid was vital to rapid and robust system integration, and explicit domain policy control permitted effective management of agent organization, behavior, security, and resources.

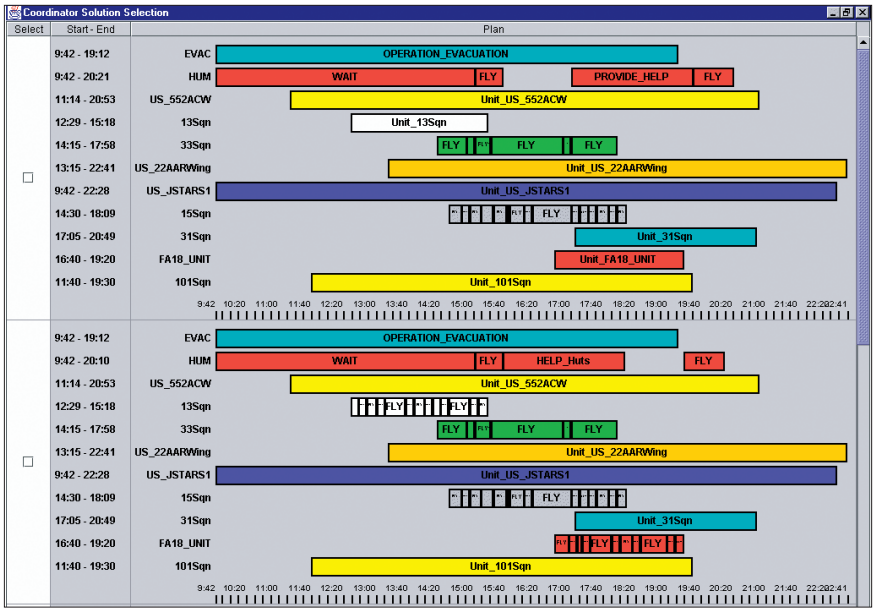


Figure 10. The Multilevel Coordination Agent removes conflicts in coalition plans. In the second solution (lower half), it has broken two missions (13Sqn and the FA18_UNIT) down to a lower level of abstraction to seek more optimal coordination.

The CoAX team's experiences show that agent-wrapping legacy systems and integrating different agent systems at short notice are relatively straightforward, and that these tasks are simpler where systems expose more of their internal information and methods. Also, heterogeneous agents can interoperate as long as implementers adhere to some minimum set of message and other standards. In fact, we believe heterogeneity should be accepted and embraced, because it is inevitable and can actually be beneficial, especially in terms of security.

Our long-term goal is to use this technology to create, support, and dynamically reconfigure virtual organizations—with coalitions being an archetypal and timely example. An agent-enabled environment creates shared understanding and improves visualization. Specific benefits accrued when agents were integrated into existing tools so as not to disrupt familiar methods of operation. The software agents operated in several roles. They worked in the background—through matchmaking, domain management, process management, and other agent services—to find, establish, and maintain the infrastructure, information, and procedural links necessary to achieve and support interoperability in a dynamically changing environment. Agents also collaborated with human operators to mediate information requests and format and display the results almost transparently.

Thus, an agent-enabled environment improves military commanders' situational awareness and could contribute significantly to network-centric warfare, an approach based on effectively linking or networking the warfighting enterprise. Indeed, cyberspace is more than an information pipe between humans; it is a battlespace in its own right. So, we should aim for "cyberspace superiority" by ensuring that coalition forces can act decisively through software agents acting on behalf of human users or mediating their actions.

We've only partially addressed the construction and maintenance of a fully dynamic virtual coalition organization. We hope to add further support for

- Adding domains and agents to the coalition structure on the fly
- Coalition partners joining or leaving unpredictably
- Handling of dynamic coalition tasks, processes, and events

Capabilities under investigation for future demonstrations include

- Obligation management to ensure agents meet their commitments
- Improved agent collaboration and runtime interoperability using Semantic Web languages and technology[13]
- Richer domain organization and security policies[5]
- Richer task, process, and event management with more dynamically determined agent relationships[14]
- Various agents providing new data types and data-processing capabilities such as threat classification and track prediction

The Fleet Battle Experiment Juliet 2002, which is part of the Millennium Challenge joint integrating experiment, will include aspects of our work.

Software agents could contribute significantly to our ability to deal effectively with unpredictable changes such as opponents' destructive activities, system failures, or withdrawn services. So far, we have shown that a software agent infrastructure can be robust and, to some extent, "self-healing." We hope to investigate this paradigm further to show that software agents can provide agility, robustness, flexibility, and functionality far beyond that provided by the individual coalition partners. ■

Acknowledgments

We gratefully acknowledge all those who contributed to the CoAX project, including Mark Burstein, Thom Bartold, Maggie Breedy, John Carson, Jeff Cox, Brad Clement, Rob Cranfill, Jeff Dalton, Pete Gerken, Bob Gray, Arne Grimstrup, Paul T. Groth, Greg Hill, Heather Holmback, Renia Jeffers, Martha Kahn, Shri Kulkarni, John Levine, Jean Oh, Pradeep Pappachan, Shahrukh Siddiqui, Jussi Stader, and Andrzej Uszok. The various projects that participated in CoAX were sponsored by DARPA and managed by the US Air Force Research Laboratory, except work by QinetiQ, which was carried out as part of the Technology Group 10 of the UK Ministry of Defence Corporate Research Programme. Bradshaw's work on KAoS policy-based agent domain services is sponsored by DARPA and the NASA Intelligent Systems and Cross-Enterprise Technology Programs.

References

1. D.S. Alberts et al., *Understanding Information-Age Warfare*, CCRP Publication Series, US Dept. of Defense Command and Control Research Program, Washington, D.C., 2001; www.dodccrp.org.

2. M. Tambe and W. Zhang, "Towards Flexible Teamwork in Persistent Teams: Extended Report," *J. Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 2, June 2000, pp. 159–183.

3. R.A. Rathmell, "A Coalition Force Scenario: 'Binni—Gateway to the Golden Bowl of Africa,'" *Proc. Int'l Workshop Knowledge-Based Planning for Coalition Forces*, Artifi-

cial Intelligence Applications Inst., Univ. of Edinburgh, Scotland, 1999, pp. 115–125; www.aiai.ed.ac.uk/project/coalition/binni.

4. J.M. Bradshaw et al., "KAoS: Toward an Industrial-Strength Generic Agent Architecture," *Software Agents*, AAAI Press/MIT Press, Cambridge, Mass., 1997, pp. 375–418.

5. J.M. Bradshaw et al., "Terraforming Cyberspace: Toward a Policy-Based Grid Infrastructure for Secure, Scalable, and Robust Execution of Java-Based Multi-Agent Systems," *Computer*, vol. 34, no. 7, July 2001, pp. 49–56.

6. T. Emerson and M. Burstein, "Development of a Constraint-Based Airlift Scheduler by Program Synthesis from Formal Specifications," *Proc. 1999 Conf. Automated Software Eng.*, IEEE CS Press, Los Alamitos, Calif., 1999.

7. M. Burstein, G. Ferguson, and J. Allen, "Integrating Agent-Based Mixed-Initiative Control with an Existing Multi-Agent Planning System," *Proc. 4th Int'l Conf. MultiAgent Systems*, IEEE CS Press, Los Alamitos, Calif., 2000; http://computer.org/proceedings/icmas/0625/0625toc.htm.

8. C.A. Knoblock and S. Minton, "The Ariadne Approach to Web-Based Information Integration," *IEEE Intelligent Systems*, vol. 13, no. 5, Sept./Oct. 1998, pp. 17–20.

9. A. Tate, J. Dalton, and J. Levine, "Generation of Multiple Qualitatively Different Plan Options," *4th Int'l Conf. AI Planning Systems* (AIPS-98), AAAI Press, Menlo Park, Calif., 1998, pp. 27–35.

10. A. Tate, "The <I-N-OVA> Constraint Model of Plans," *Proc. 3rd Int'l Conf. AI Planning Systems*, AAAI Press, Menlo Park, Calif., 1996, pp. 221–228.

11. J. Fraser and A. Tate, "The Enterprise Tool Set—An Open Enterprise Architecture," *Proc. Workshop Intelligent Manufacturing Systems, Int'l Joint Conf. Artificial Intelligence* (IJCAI-95), Morgan Kaufmann, San Francisco, 1995, pp. 95–103.

12. B.J. Clement and E.H. Durfee, "Top-Down Search for Coordinating the Hierarchical Plans of Multiple Agents," *Proc. 3rd Int'l Conf. Autonomous Agents*, ACM, New York, 1999, pp. 252–259.

13. D.N. Allsopp et al., "Toward Semantic Interoperability in Agent-Based Coalition Command Systems," *Proc. 1st Int'l Semantic Web Working Symp.*; www.semanticweb.org/SWWS.

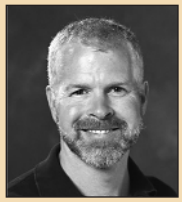
14. A. Tate, J. Dalton, and J. Stader, "I-P^2—Intelligent Process Panels to Support Coalition Operations," *Proc. 2nd Int'l Conf. Knowledge Systems for Coalition Operations* (KSCO-2002), Artificial Intelligence Applications Inst., Univ. of Edinburgh, Scotland, 2002, pp. 184–190; www.aiai.ed.ac.uk/project/coalition/KSCO.

The Authors

David N. Allsopp is a researcher in the Distributed Technology Group at QinetiQ. His current research interests include software agents in support of command information systems, mixed-initiative planning systems, and Semantic Web technologies. He received a BA in natural sciences and a PhD in materials science from St. John's College, University of Cambridge. Contact him at QinetiQ, Malvern Technology Centre, St. Andrews Road, Malvern, Worcestershire, WR14 3PS, UK; d.allsopp@signal.QinetiQ.com.

Patrick Beautement is a principal scientist in the QinetiQ Sensors and Electronics Sector Distributed Technology Group, where he is the technical leader of the Intelligent Distributed Systems initiative. He is a member of the DARPA CoABS/CoAX principal investigators team, and he helped develop the Effects-Based Wargaming concept. He was previously a squadron leader in the Royal Air Force and a wargame systems specialist at the Air Warfare Centre. He has a master's degree in intelligent systems. Contact him at QinetiQ, Malvern Technology Centre, St. Andrews Road, Malvern, Worcestershire, WR14 3PS, UK; Patrick.Beautement@signal.QinetiQ.com.

Jeffrey M. Bradshaw's biography appears in the guest editor introduction on page 16.



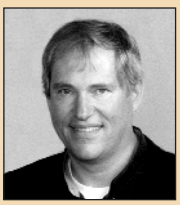
Edmund H. Durfee is a professor of electrical engineering and computer science and of information at the University of Michigan. His teaching and research focus on artificial intelligence, multiagent systems, and real-time intelligent control. He is a senior member of IEEE and an AAAI fellow. He received a PhD in computer science from the University of Massachusetts. Contact him at the EECS Department, University of Michigan, 1101 Beal Ave., Ann Arbor, MI 48109; durfee@umich.edu.

Michael Kirton is a QinetiQ fellow in the Distributed Technology Group, QinetiQ. His current research focuses on agent, grid, and Semantic Web technologies. He is a member of the UK e-Science Core Programme Technical Advisory Group. He received a BSc in physics and a PhD in theoretical physics from the University of Newcastle upon Tyne. Contact him at QinetiQ, Malvern Technology Centre, St. Andrews Road, Malvern, Worcestershire, WR14 3PS, UK; m.kirton@signal.QinetiQ.com.

Craig A. Knoblock is a senior project leader at the Information Sciences Institute and a research associate professor in computer science at the University of Southern California, where he leads the USC Information Agents Research Group. His research interests include information agents, information integration, automated planning, and machine learning. He is a member of the AAAI Executive Council. He received his PhD in computer science from Carnegie Mellon University. Contact him at the University of Southern California Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292; knoblock@isi.edu.

Niranjan Suri is a research scientist at the Institute for Human and Machine Cognition at the University of West Florida, where he also teaches computer science. His research interests include software agents, mobile agents, mobile and ubiquitous computing, virtual machines, and network and system security. His recent research led to development of the Nomads mobile agent system for Java-based agents. He developed and implemented a Java-compatible Virtual Machine, Aroma-VM, with key extensions to support mobile state and resource control. Contact him at the Institute for Human and Machine Cognition, University of West Florida, 40 S. Alcaniz St., Pensacola, FL 32501; nsuri@ai.uwf.edu.

Austin Tate's biography appears in the guest editor introduction on page 16.



Craig W. Thompson is cofounder of Object Services and Consulting. He was editor of OMG's *Object Services Architecture*, organized the OMG Agent Special Interest Group, and holds six software patents with one pending. An IEEE senior member, he received a PhD in computer science from the University of Texas at Austin. Contact him at Object Services and Consulting, 2725 Deep Valley Trail, Plano, TX 75023; thompson@objs.com.