Working Notes of the AAAI Fall Symposium on Negotiation Methods for Autonomous Cooperative Systems
North Falmouth, Massachusetts

Multi-Agent Mission Coordination via Negotiation

David J. Musliner and Kurt D. Krebsbach
Automated Reasoning Group
Honeywell Laboratories
3660 Technology Drive
Minneapolis, MN 55418
{musliner,krebsbac}@htc.honeywell.com

Introduction

This paper is intended to give an intuitive overview of the operations of MASA-CIRCA, the Multi-Agent Self-Adaptive Cooperative Intelligent Real-Time Control Architecture. While individual CIRCA agents have been under development for some time, we have only recently begun developing the architecture's multi-agent capabilities. This paper briefly describes the high-level negotiation functions that MASA-CIRCA currently uses to coordinate multiple agents, in the context of an implemented demonstration scenario.

As an architecture for autonomous control, CIRCA is distinguished by its strong emphasis on real-time performance guarantees. Individual CIRCA agents, as illustrated in Figure 1, combine two levels of planning and automatic controller synthesis modules with a plan executive (the Real-Time Subsystem) that is responsible for reactively executing automatically-generated control rules in hard real time. CIRCA is designed to adapt, on the fly, to changes in its environment and its capabilities by building and executing new reactive plans. Because CIRCA reasons explicitly about the timing constraints that its reactive plans must meet, and because it plan executive provides rigidly predictable performance, CIRCA supports the performance guarantees required for autonomous systems applications in mission-critical real-time domains.

In the new multi-agent versions of CIRCA, we are extending these real-time performance characteristics to a team of coordinating CIRCA agents, each controlling a separate platform (e.g., a team of unmanned combat air vehicles, or UCAVs). As shown in Figure 2, the CIRCA agents communicate and negotiate at all levels of the architecture to coordinate their activities.

The AMP is responsible for the highest-level con-

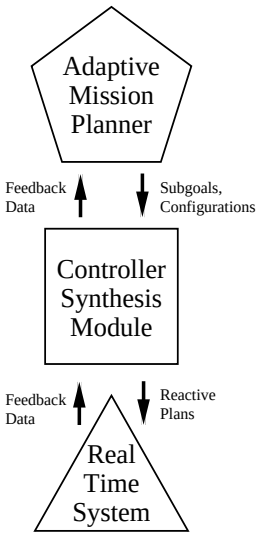

Figure 1: Abstracted view of a single CIRCA agent that provides real-time planning and control.

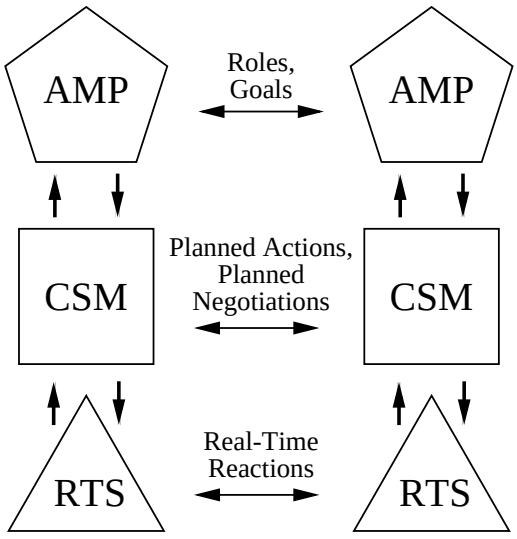

Figure 2: Two CIRCA agents negotiate to dynamically allocate team resources.

trol of a CIRCA agent (Musliner, Durfee, & Shin 1995; Musliner *et al.* 1999), managing the agent's responsibilities (by negotiating with other agents) and the agent's deliberation activity. The AMP performs intelligent allocation of deliberation effort via a negotiation and tradeoff process with the Controller Synthesis Module (CSM) (Musliner 2000). Our current demonstration illustrates early forms of the coordination and negotiation that occurs between AMPs. Before describing this demonstration, we provide an overview of the AMP's operation and brief introduction to the display systems used to illustrate CIRCA's operations.

Adaptive Mission Planner Overview

At the highest level, the AMP's primary responsibility is managing an individual agent's tasks and coordinating with other agents to achieve the overall team mission. The AMP does this by determining what tasks are its responsibilities through negotiation with other cooperating agents, and then arranging to have plans (controllers) generated to successfully address those tasks during the execution of the mission.

The overall team mission is divided into *phases*, which correspond to modes or time intervals that share a fundamental set of common goals, threats, and dynamics. For example, our UCAV scenarios include missions that have phases such as ingress, attack, and egress. The ingress phase is distinguished from the attack phase both by the characteristics of the flight path (e.g., a nap-of-earth stealthy approach vs. a popup maneuver very near a target) and by the expected threats (e.g., the types of missile threats present at different altitudes) and goals (e.g., reaching the target zone vs. deploying a weapon).

In this context, the team of CIRCA agents must arrange to have different agents responsible for different goals and threats, depending on their available capabilities and resources (e.g., ECM equipment and weapons loadout). For each mission phase, the CIRCA agents must have plans, or controllers, that are custom-designed (either before or during mission execution) to execute the mission phase and make the best possible effort to achieve the goals and defeat the threats associated with the phase. The CSM, described elsewhere (Musliner, Durfee, & Shin 1993; 1995), is capable of automatically building these controllers, but this controller synthesis can be a complex and time-consuming process. The complexity (and hence duration) of the CSM process can be controlled by varying the *problem configuration* that is passed to the CSM to describe the characteristics of the desired controller for a particular mission phase (Musliner 2000).

The AMP is thus also responsible for determining which mission phase the CSM is trying to build a controller for at any moment, and how hard it should work to do so, by modifying the phase problem configurations. This is what we call the AMP's *deliberation scheduling* function. In the current implementation, this corresponds to altering which of the phase's potential goals and threats should be considered. In future versions, even more flexible techniques will be available to adjust the CSM problem-solving complexity.

Given a problem configuration, the CSM projects future world states and plans actions that keep the system safe and direct it towards its goals. The CSM also computes deadlines on these actions, indicating how quickly they must occur after a particular world state is reached, to avoid catastrophic failures (e.g., destruction by a threatening surface-to-air missile). The CSM reduces these action plans to cyclic schedules of Test-Action Pairs (TAPs), which test the state of the world and take a chosen action. The RTS can then execute these TAP schedules, predictably enforcing the timing constraints on reaction planned by the CSM to keep the system safe and achieve its goals.

Inter-Agent Negotiation

Leveraging significant amounts of existing code that we wrote for an internally-funded project on distributed scheduling, we have already built and demonstrated multiple CIRCA AMPs negotiating over the allocation of mission-level responsibilities (e.g., handling missile threats). Using a Contract-Net-like arrangement, the AMPs submit bids to handle these responsibilities (Smith 1977). Currently the computation of bid values is artificial, but eventually it will reflect the expected costs and benefits that an agent expects to incur if it assumes a particular responsibility. For example, an agent might assign a certain value to planning to use its electronic countermeasures (ECM) to defeat certain types of radar-guided missile threats.

Each contract goes through a series of modes as it is processed by the system:

New — New contracts are formed when an agent learns of a potential threat or goal for a particular mission phase. In the current implementation, only the `MASTER` learns of new threats and goals. This simplification was made purely for convenience, and is not constrained by the current infrastructure. The agent that learns of the new threat or goal forms a new contract object and broadcasts an announcement of its contents to all the agents. The agent is termed the contractor agent for that contract.

Announced — Once a contract is announced, each agent is responsible for replying with a bid that is used to determine which agent will assume responsibility for the contract. Currently, bids are essentially pre-specified in the domain description that indicates which agent can accomplish which types of tasks. For example, a domain description may say that a certain agent is highly capable of handling a particular type of threat, and it should bid a certain constant value (e.g., 10). In the future, the bid will be computed by more complex methods that assess the agent's capabilities, physical resources, load level, and ability to compute a new controller in time.

Awarded — Once the contractor agent receives all of the bids for a particular contract, it awards the contract to the highest bidder, notifies the other agents they are not the winner, and updates the contract status.

Planned-for — The agent that is awarded a new contract is responsible for building a new controller that can handle the corresponding new threat or goal. When the controller synthesis process is complete, the contract status is set to `:planned-for`.

Completed — Each threat or goal is specific to a mission phase. When that mission phase has been completed (e.g., the aircraft has completed the attack phase), the associated threat/goal contracts are marked `:completed`.

Failed — If an agent fails to generate a plan for a contract, or it is disabled and thus unable to execute the controllers it has generated, the contract is marked failed and the contractor agent re-announces it for new bids. The re-negotiation proceeds just as with the original process, so that a new agent is awarded the contract and builds a controller to handle it.

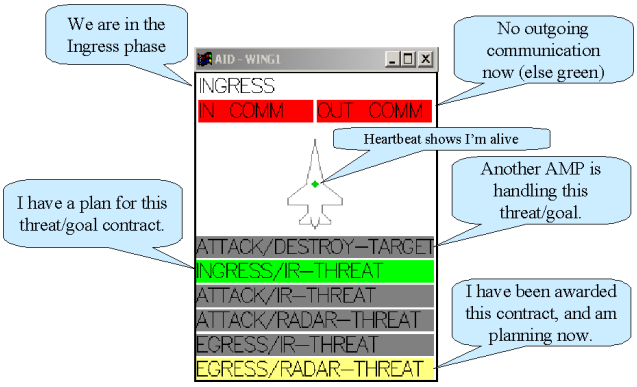

Figure 3: The AMP Information Display depicts AMP status.

Currently, the AMPs can negotiate over contracts that represent responsibility for threats and goals. When awarded a contract for a threat or goal, the AMP can then run the CSM to synthesize customized controllers using CSM problem configurations that are automatically generated to describe the negotiated responsibilities.

In addition, the AMPs can handle messages saying that another AMP has died[1], and will issue contracts to re-negotiate the responsibilities formerly held by the dead agent. The AMP maintains data structures to generate and track multiple CSM problem configurations for each mission phase, and it can store the associated controllers generated by the CSM. In addition, the AMP can download the controllers to the RTS as soon as they are available.

AMP Information Display

To provide a compact graphical indication of what functions the AMP is performing, we have developed an AMP Information Display (AID). The AMP sends its status information to the AID over a socket. Illustrated in Figure 3, the AID gives the observer visibility into the multi-agent negotiation and planning processes.

At the top of the AID, two labels indicate which CIRCA agent the display is describing, and what mission phase that agent is currently executing. Below

[1]The underlying socket communications layer inherently recognizes the loss of connection and returns an error.

those lines, two light bars labeled "IN COMM" and "OUT COMM" flash when the agent is receiving or sending over its socket connections to other agents. Further down, the aircraft icon will display the status of various aircraft subsystems. This aspect of the AID is obviously domain-dependent, and it is not fully implemented yet. In the future, we plan to add iconic representations of engines, defensive systems, and other subsystems subject to damage and imperfect operation. Currently, the only functioning element of the iconic display is the center diamond, which acts as a "heartbeat" for the AMP, blinking periodically when the AMP is idle but still functioning.

Below the iconic display, a set of lines display the status of each of the contracts in the overall multi-agent system, according to the single CIRCA agent's view. Each contract represents a threat or goal that must be handled in a particular mission phase. Each line has a text label identifying the contract it represents, and a color (not apparent in black-and-white printouts) that describes the status of the contract:

Red indicates that a new threat/goal contract has arrived and is not yet even announced for bids by other agents.

Pink indicates that a new contract has been announced for bids, but all the bids have not yet been received.

Yellow indicates that this agent has been awarded this contract (i.e., it has accepted responsibility to "handle" this goal or threat).

Green indicates that this agent has successfully generated a new controller (plan) to handle this contract.

Gray indicates that another agent has responsibility for this contract.

Purple indicates that this agent was previously responsible for this contract, but is now disabled and will not handle the associated threat/goal.

As the AMP grows in capabilities, especially in its ability to make tradeoffs between deliberation time and the expected performance of synthesized controllers, we will extend the AID to dynamically visualize these tradeoffs and give the observer some insight into the operations of the AMP. For example, we expect to have gauges, graphs, or other intuitive graphical representations showing performance characteristics such as the likelihood of failure for different mission phases using the current-best controllers, etc.

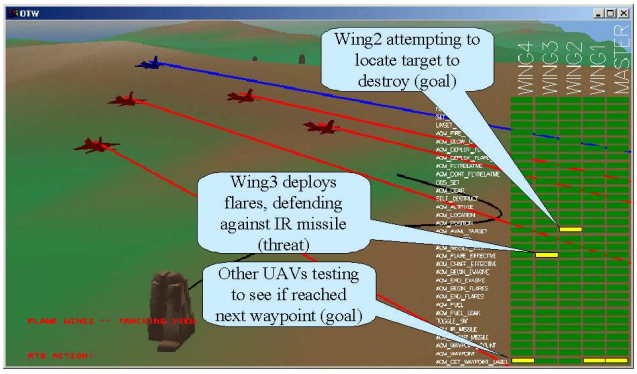

Figure 4: The RTS Information Display shows what each RTS is doing at any time.

RTS Information Display

The RTS can execute TAPs at extremely high speeds, so it is very difficult for an observer to follow its runtime activities. However, it can be very useful to have *some* indications of what the RTS is doing. To that end, we have developed an RTS Information Display (RID). As illustrated in Figure 4, the RID uses flashing bars of color to indicate each primitive test or action the RTS executes. As these color bars flicker rapidly on and off, the observer can recognize different patterns of RTS activity, and can also discern individual distinct actions that are infrequent. If a pattern of RTS activity persists for a few seconds, the observer can also identify specific activities the RTS is performing, by mapping the flashing bars back to the labels on the left edge of the display.

Recall that the AMP sends its status information to the AID over a socket. We could have designed the RTS/RID interactions in a similar way. However, because that socket communication could add significant overhead to the RTS operations, we have chosen instead to build this version of the RID directly into the flight simulation system used for our demonstration. This avoids both additional communication for the RTS and additional screen real-estate: the RID is drawn as a "stencil" over the top of the simulation display.

This approach has one notable weakness: the RID can only display RTS tests and actions that are sent to

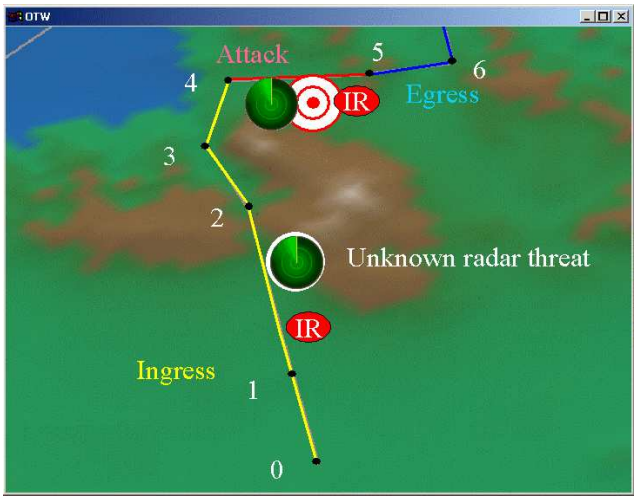

Figure 5: This mission overlay shows the expected path with known and unknown threats and targets.

the simulator (i.e., those that interact with the world outside of CIRCA); it cannot show the RTS activities that are purely internal. For example, several TAPs in every TAP schedule are dedicated to downloading new TAP schedules, switching between TAP schedules when a suitable state is reached, etc. These "internal" TAPs cannot be visualized on the RID embedded inside the simulator.

For situations where visualizing these internal activities is also important, we have a different version of the RID available that runs in a standalone mode separately from the simulation environment, but is currently only compatible with Xwindows displays. This version can show all RTS primitives calls, including internal functions such as reading in new TAP schedules (controllers).

Demonstration Scenario

Goal

The primary goal of this demonstration scenario is to illustrate negotiation between CIRCA agents and dynamic generation of controllers on the fly. The demonstration shows the agents negotiating diverse responsibilities and synthesizing customized controllers at the start of the mission, as well as re-negotiating roles and dynamically building new controllers during the mission when unexpected circumstances arise.

Mission

The demonstration scenario contains a flight of five unmanned F16-type fighter aircraft (`MASTER` and `WING1` through `WING4`) whose mission is to destroy a ground target while defending themselves against attack. Figure 5 shows a top-down view of the planned mission flight path, along with threat and goal location icons. The mission consists of three phases which correspond to three distinguished flight path segments: ingress, attack, and egress. During the mission, the AMPs need to create plans that account for several goals and expected threats:

1. Defending against IR-guided missile threats during the ingress phase.
2. Defending against IR-guided and radar-guided missile threats during the attack phase.
3. Destroying the target during the attack phase.
4. Defending against IR-guided missile threats during the egress phase.

In addition, one unexpected threat is displayed in Figure 5: a radar-guided missile site along the ingress path that is not reported to the CIRCA agents (presumably because it is unknown to our forces). This unexpected threat will turn out to be fatal to `WING4`, and this leads to the re-negotiation and replanning activity.

Demonstration Storyboard

When the mission begins, the fighters negotiate responsibilities for the first time. The `MASTER` is tasked with defending against radar-guided SAM sites during the attack phase of the mission. `WING1` gets this responsibility during the egress phase. IR threats are handled by `WING1` during ingress, `WING3` during attack, and `WING4` during egress. Responsibility for destroying the target is given to `WING4`. Once plans are constructed for the initial phases, the aircraft begin flying the mission (even before controllers for the latter phases are complete).

Before the aircraft leave the runway, controllers have been generated for all of the mission phases and the AID shows green lights for all of the goal and threat contracts. Then, shortly after the aircraft pass over their first waypoint, the anticipated IR threat attacks. Figure 4 illustrates the scene where `WING1` has been watching for IR threats and has begun deploying flares to confuse the incoming IR-guided missile. By repeat-

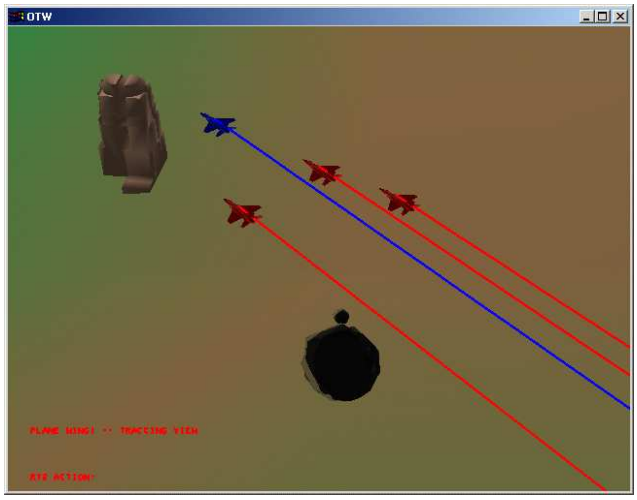

Figure 6: `WING4` exploding.

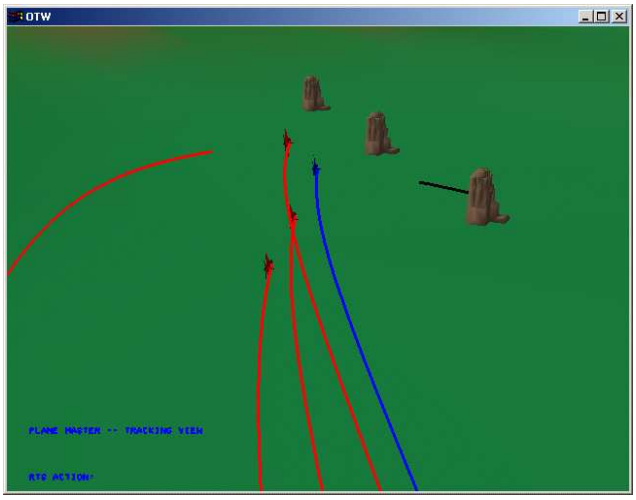

Figure 7: After `WING4` dies, the surviving CIRCA agents re-negotiate and generate new plans to handle its responsibilities and ensure mission success. Moments later, as the attack missile arcs in, the CIRCA team takes evasive maneuvers to avoid a rising radar-guided missile.

edly launching flares until the missile explodes on one, `WING1` successfully defeats the IR-guided missile.

A short time later the aircraft pass near the unexpected radar-guided missile site, which attacks. Unfortunately, since we did not tell the CIRCA agents about this potential threat, they have not built controllers that look for this danger. Unawares and unresponsive, `WING4` is destroyed (see Figure 6).

The AMPs immediately detect that `WING4` is dead, and re-negotiate its contract responsibilities. `WING2` gets the contract for destroying the target and `WING3` gets the egress IR threat. Within three seconds the re-negotiation process *and* the new CSM invocations are complete, and the agents have created new controllers to handle their altered responsibilities. With all contract lights green (as shown in Figure 7), the team has recovered and the mission is still headed for success.

The aircraft continue to fly along the ingress flight route to waypoint four, where they enter the attack phase. When the target is in range, `WING2` fires a surface-to-ground missile. While that attack missile is on its way to the target, the fighters are threatened again (as illustrated in Figure 7). This time, the `MASTER` leads the flight into evasive maneuvers that defeat the radar-guided SAM. Figure 7 shows the aircraft at this time. During the evasive maneuvers, the second SAM site near the target launches an IR-guided missile at the team. In response, `WING3` deploys flares to defeat that missile. The aircraft then proceed safely to waypoint five. At waypoint five, the flight enters the egress phase of flight, the CIRCA agents begin executing a different set of controllers that are concerned about a different set of threats and goals, and the scenario proceeds without further incident.

Related Work

A wide variety of prior work exists on building multi-agent systems that negotiate to coordinate their behavior. As noted earlier, we have used the Contract Net (Smith 1977) approach almost without change, so we are not claiming a unique contribution in negotiation methods. Rather, we hope to apply existing methods (and develop new ones as necessary) in a novel context: the hard real-time domains, online controller synthesis, and performance guarantees of CIRCA. The initial demonstration described above has accomplished part of our objective. While we demonstrated inter-agent negotiation that impacts CIRCA's online controller synthesis and runtime performance guarantees, that negotiation itself has not yet been subjected to the real-time restrictions and AMP deliberation scheduling paradigms (Goldman, Musliner, & Krebsbach 2001) we are developing. In the future, we expect that negotiation with other agents will be just another tool in the AMP's kit of possible approaches to trading off

performance and safety (Musliner 2000). When unable to build a fully-guaranteed controller for all of the goals and possible threats in a particular mission phase, the AMP may choose to negotiate with other agents to offload some responsibilities. The time that this negotiation itself takes will be considered in deciding whether distributed or purely local performance tradeoffs should be attempted.

Acknowledgments

This material is based upon work supported by DARPA/ITO and the Air Force Research Laboratory under Contract No. F30602-00-C-0017. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, the U.S. Government, or the Air Force Research Laboratory. The demonstration described in this paper is the result of work with colleagues including Jeff Rye and Robert Goldman.

References

Goldman, R. P.; Musliner, D. J.; and Krebsbach, K. D. 2001. Managing online self-adaptation in real-time environments. In *Proc. Second International Workshop on Self Adaptive Software*.

Musliner, D. J.; Goldman, R. P.; Pelican, M. J.; and Krebsbach, K. D. 1999. Self-adaptive software for hard real-time environments. *IEEE Intelligent Systems* 14(4):23–29.

Musliner, D. J.; Durfee, E. H.; and Shin, K. G. 1993. CIRCA: a cooperative intelligent real-time control architecture. *IEEE Trans. Systems, Man, and Cybernetics* 23(6):1561–1574.

Musliner, D. J.; Durfee, E. H.; and Shin, K. G. 1995. World modeling for the dynamic construction of real-time control plans. *Artificial Intelligence* 74(1):83–127.

Musliner, D. J. 2000. Imposing real-time constraints on self-adaptive controller synthesis. In *Proc. Int'l Workshop on Self-Adaptive Software*.

Smith, R. 1977. The contract net: A formalism for the control of distributed problem solving. In *Proc. Int'l Joint Conf. on Artificial Intelligence*, volume 1, 472.