Real-time Control of Intelligent Agents

Paul Scerri[1] and **Nancy Reed**[2]

Abstract. In this demonstration we show a prototype system that allows users to issue commands to simulated actors in real-time. The interactive, real-time control is a form of *Adjustable Autonomy* [1, 2]. The real time control prototype, nicknamed "The Boss" is a sub-system of a complete actor specification system called EASE (End-user Actor Specification Environment). The Boss allows tasks, at varying levels of abstraction, to be assigned, withdrawn or suspended while an agent continues running. These facilities together provide the user a large amount of control over the actor at runtime. Importantly the user is not required to take back complete control of the actor in order to achieve some particular task. The Boss works by adding agents to, removing agents from or suspending agents in the multi-agent system (MAS) which in turn controls the actor. Because the addition and removal of agents to the multi-agent system happens continually even without user interaction, the overall behavior of the simulated actor is smoothly integrates a user's adhoc commands.

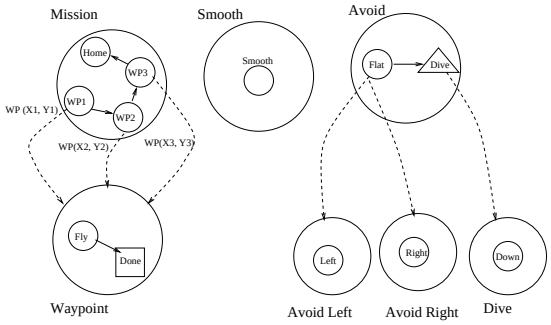

Figure 1. An example of an EASE MAS specification for a simple patrol actor for the air-combat domain. Large circles show agents, small circles show normal states, triangles show failure states and squares success states. Arrows indicate transitions between states. Dashed lines indicated contractual relationships.

1 EASE

EASE is a set of graphical development tools for building intelligent actors for simulation environments. Specific tools support all aspects of development from information processing specification through to mission specification. For more details on EASE see [4].

An actor controlled by EASE has an entire multi-agent system (MAS) for the task of action selection. Each agent in the system is responsible for a specific aspect of the actor's overall behavior. The agents form contracts with other agents into a hierarchical structure. The *contracted* agent is assigned a specific aspect of the *contractor*'s tasks. The agents at the bottom of the hierarchy negotiate amongst themselves about the next action of the actor. Each of the agents in the negotiation "argues" for actions that would best achieve the specific task it has been assigned, but can "compromise" and if required it will accept values that will only partially achieve the task it has been assigned.

A *factory* administers a negotiation. A factory continuously selects actions from the domain of possible actions and suggests selected values to the agents negotiating with the factory. The interested agents determine their *satisfaction* with the suggested action and return the satisfaction value to the factory. The factory's algorithm for choosing between actions is an anytime version of a fuzzy behavior fusion algorithm [3].

In the current implementation of EASE the behavior of individual agents is governed by a simple state-machine. In each state the agent will form contracts or negotiate for certain types of actions. Two special types of states and two special types of transitions between states can be part of a state-machine. When in a *failure* state the agent is indicating to its contractor that it is currently unable to achieve its required task. In a *success* state the agent is indicating to its contractor that it is succeeding (in the case of a XX task) or has succeeded (in the case of a YY task). A *failure transition* will be taken in a contractor agent when *any* of the agents it has contracted are in failure states, similarly a *success transition* is taken when *all* contracted agents are in success states. The actor designer is responsible for defining the success and failure states and transitions.

A design-time view of an agent hierarchy for a simple patrol mission in the air combat domain is shown in figure 1. When the actor is started there are three agents active: the mission agent (that controls the overall mission); smooth agent (which prevents the pilot from trying too extreme manouvres); and the avoid agent (which keeps the pilot away from obstacles). As the mission progresses other agents will be contracted for flying to particular waypoints and avoiding particular obstacles.

2 The Boss

To effectively control an actor at runtime a user needs two tools : a mechanism for viewing the current state of the actor and a mechanism for exerting control. The former is important mainly for determining reasons for incorrect, undesired or unexpected behavior, the latter for actually giving commands to the actor.

2.1 Observing the Actor's multi-agent system

The primary tool for observing the actor's controlling MAS is shown in Figure 2. It uses a familiar tree layout to show the currently active

[1] Real-time Systems Laboratory, Department of Computer and Information Science, Linköping University, SE-581 83 Linköping, Sweden, pausc@ida.liu.se
[2] Same address as above. nanre@ida.liu.se

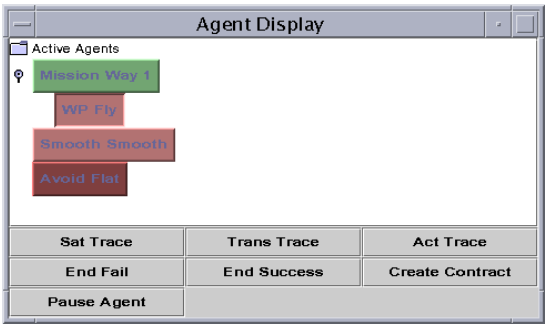

Figure 2. The Boss display lists all active agents in an actor (top) and allows tracing, termination and suspension of a selected agent (bottom).

agents and the hierarchy of contracts. Each node in the tree shows the name and current state of one of the agents. The tree shows how a particular abstract task is currently broken down into sub tasks.

The top three buttons in the bottom of The Boss window (Figure 2) allow the user to view different aspects of individual agents in more detail. For agents involved in negotiations, a snap shot of the satisfaction function can be shown (Sat Trace button). The snapshot shows the calculation the agent has performed in order to produce its satisfaction value for the current factory suggestion. For all the agents, traces of the calculations of their priority (Act Trace) and any transition conditions (Trans Trace) can be shown. These snapshots allow a user to investigate why an agent is persuing a particular course of action. When the user believes an agent is doing something wrong they can check the agents calculations in detail to see whether there are errors or ommissions in the agents reasoning or whether the agent has, correctly, taken into account factors that the user has not.

2.2 Manipulating the Actor's multi-agent system

There are currently three ways that the user can manipulate the MAS: by breaking existing contracts, by creating new contracts and by suspending existing agents. Each of the mechanisms is accessed through the same interface (Figure 2).

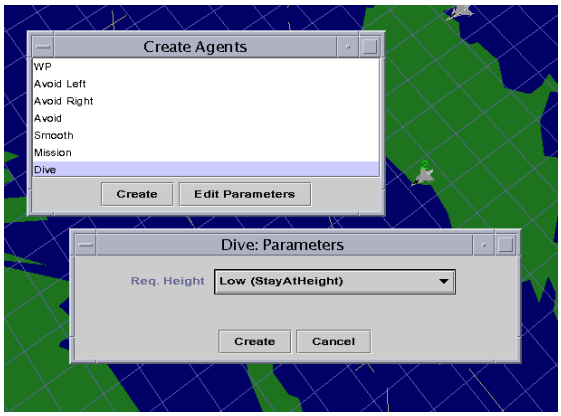

Figure 3. Creating new agents

To break a contract with an agent the user selects the agent and then selects either the "end fail" button or the "end success" button at the bottom of the screen. The agent whose contract the user has broken in turn ends contracts with any agents it has contracted, i.e. the user can end a whole hierarchy of tasks with a single click. If the "end failure" button was selected the agent then goes into a dummy failure state, otherwise it goes into a dummy success state. A contractor that contracted the agent whose contract the user has broken handles the termination of the contract as it would handle any success or failure of a contractee.

For example, consider the example actor specification given in the previous section (shown in Figure 1). If a user decided that the aircraft should skip the first way-point and go directly to the second, they could achieve their objective by selecting the waypoint agent (contracted by the mission agent) then selecting "end success". The waypoint agent will leave the negotiations and enter a dummy success state. The mission agent (i.e. the waypoint agent's contractor) will detect that the way-point agent is now in a success state and will contract a new way-point agent for the next waypoint.

The second way a user can manipulate the MAS is by creating new agents. Figure 3 shows another part of The Boss which allows agents to be created and contracted by the user. To create an agent, the user selects the agent type, then selects "create". If the agent contract requires parameters to be instantiated (for example the location of a waypoint) a dialog box pops up allowing the user to instantiate the necessary parameters. The created agent enters the MAS and immediately makes contracts or joins in negotiations as specified in its starting state. This functionality effectively allows the user to dynamically task the actor. In early evaluations of The Boss this functionality has been found to be particularly useful in testing actors.

The final type of control the user has is the ability to suspend (pause) an agent within the actor. This is done by selecting the agent and then selecting the "pause agent" button shown at the bottom of The Boss display (Figure 2.) As with the contract breaking mechanism if an agent at the top of a hierarchy is paused the whole hierarchy is paused. This mechanism allows the user to indicate to the actor that a particular task should be paused.

3 Acknowledgments

This work is supported by Saab Corporation, Operational Analysis Division, the Swedish National Board for Industrial and Technical Development (NUTEK) under grants IK1P-97-09677, IK1P-98-06280 and IK1P-99-6166, and the Center for Industrial Information Technology (CENIIT) under grant 99.7.

REFERENCES

[1] Henry Hexmoor, editor. *Workshop on Autonomy Control Software*. Autonomous Agents 1999, May 1999.

[2] D. Kortenkamp, G. Dorias, and K. Myers, editors. *Proceedings of IJCAI99 Workshop on Adjustable Autonomy Systems*, August 1999.

[3] Paolo Pirjanian and Maja Mataric. A decision theoretic approach to fuzzy behavior coordination. In *Proceedings, IEEE International Symposium on Computational Intelligence in Robotics & Automation (CIRA-99)*, Monteray, CA, Nov. 1999.

[4] Paul Scerri and Nancy Reed. Creating complex actors with EASE. In *Proceedings of Autonomous Agents 2000*, 2000. Poster Presentation.