

Headless Chickens IV

Paul Scerri, Nancy Reed, Tobias Wiren, Mikael Lönneberg and Pelle Nilsson

Institute for Computer Science, Linköpings Universitet

1 Introduction

The Headless Chickens IV (HCIV) were developed using a prototype end-user agent development environment called EASE (End-user Actor Specification Environment). The experience provided a great deal of data about general and domain specific properties of the system. A secondary research goal was to generalize the strategy editor used to specify the team strategies of last years team to specify team strategies for agent of different types via the use of an XML interface. The poor performance of the team (3 losses and a draw in the competition) is attributed to the early stage of the research as well as the very high computational complexity of the agent runtime architecture.

2 Special Team Features

In 2000 the aim of the work on the Headless Chickens RoboCup soccer team was to use an existing end-user actor development system in a second domain. EASE is a prototype system developed to allow relative end-users to specify the behavior of intelligent agents for interactive simulation environments[7]. Until now EASE has been primarily used for developing air-combat agents for a simulation environment called TACSI. Using EASE in a second domain allowed us to learn more about the strengths and weaknesses of the underlying architecture and, in particular, learn which ideas could be generalized and which could not.

The EASE runtime engine is essentially a behavior-based architecture[3, 1] with a few enhancements to make it easier for relative end-users to specify the behavior of complex actors in the EASE development environment. Development of players is completely graphical once a few Java classes have been created to interface EASE to the simulation environment.

One of the key features of EASE is an interactive view of the internal reasoning of a player. The viewer, nick-named *The Boss*, not only shows the hierarchical arrangement of behaviors in the actor at runtime but also allows users to start, stop and pause behaviors at runtime[8]. This means that while testing an agent the user can interactively manipulate the agent's internal reasoning to determine the effect of different potential specification changes. Also, the user can interactively get the agent to carry out specific tasks at runtime even though the agent as specified would not have done that task.

During the development of the HCIV, *The Boss* was found to be very useful for helping to incrementally develop and debug players. For example, when the

players were doing something wrong behaviors could be started one at a time allowing the developer to determine which behavior(s) was causing the problem. Also the developer could work effectively with incompletely specified players by interactively issuing commands, for example if no strategies had yet been defined for handling free kicks when a free-kick occurred the developer could interactively direct a player to chase and kick the ball.

Behavior-based architectures use a variety of methods to determine a final agent action given a number of competing behaviors[6]. EASE uses a method based on an algorithm by Pirjanian[4], primarily designed for robots. Pirjanian's algorithm relies on checking all possible actions, albeit in an efficient manner. In real-time interactive simulations running on normal hardware such an approach is unacceptably slow. EASE uses a probabilistic, anytime algorithm[10] to find reasonable actions quickly then incrementally find better actions, if time permits. One property of the algorithm is that when the situation changes dramatically it may take longer than one cycle for the agent to determine the optimal action, but it will take whatever action it has found after the length of one cycle. For air-combat simulation this is acceptable as actions usually involve picking a new heading or speed which can be subsequently slightly adjusted as better headings or speeds are found, e.g. in the first cycle selecting a heading of 127° then in the next cycle finding a slightly better heading of 125° before finally settling on 123° . Such adjustments are not noticed as the turn will usually take many cycles to execute. However, the algorithm ran into problems in the RoboCup domain when trying to kick the ball. The first action the agent takes changes the situation dramatically not allowing for slowly improving the action. Until some "hacks" were introduced the agent would often kick the ball in very strange directions or with strange power values – simply because the action determination algorithm did not have enough time to find a better action. This experience has led us to look to more sophisticated probabilistic techniques such as simulated annealing or genetic algorithms for searching for a good action[2, 5].

3 World Model

The HCIV world model saves very little information from cycle to cycle and does very little reasoning on the information it has. The information that is stored across cycles is usually in relation to objects that may not be seen in the next cycle. The simplicity of the world model is primarily due to the implementation being so new, not any design philosophy.

A feature of the world model is a viewer to show at runtime of most of the agent's calculations, in intimate detail without the designer having to embed debugging code. The *Calculation Debugger* provides a useful tool for investigating problems with the agents behavior very quickly. However, the Calculation Debugger relies on a very computationally expensive method of doing calculations which slows a agent's overall behavior.

4 Communication

There is no communication between players. Each player knows the situations that will lead to different team strategies and the triggers for changing strategies are designed to minimize confusion, e.g. by using general ball position or referee calls. Communication might be used to improve the players model of the world, however so far we have found that when accurate information is required, for example when dribbling, the relevant player has best access to that information. In cases when other players have the best information it is usually not so important that the player has very accurate information. One exception to this would be for passing but our team strategy of passing to positions from the pre-defined strategies make this irrelevant.

5 Skills

The skills of the players are completely specified within the EASE graphical development environment. In the RoboCup domain, building skills in EASE has a significant advantage and a significant disadvantage over programming in low level code. The main advantage is that EASE enforces a good functional breakdown where each function must be in a separate behavior. For example dribbling the ball and obstacle avoidance will be in separate behaviors. The behavior fusion algorithm automatically combines the two functions. On the negative side graphical specification was not well suited to the types of calculation that needed to be done in player skills (a mathematical specification style may have been better suited).

6 Strategy

A secondary research goal of the HCIV development was to test the generality of the team level strategy editor developed as part of the 1999 team (HCIII). The strategy editor allows high level specification of team strategies and has been found to be very useful[9]. In 1999 the editor was closely coupled to a particular behavior based architecture that was being used. This year an XML import and export interface was created for the editor allowing a variety of different agents to use the high level strategy specification. EASE agents (i.e. HCIV agents) as well as HCIII agents worked with the same high level specification.

The XML files can be read (and interpreted) by a variety of different agent types. The input XML files specify the modes of play that the agents know about (e.g. defensive corner kick), the types of actions that the agents can take in each mode (e.g. pass and dribble) and the available styles of play (e.g. defensive or attacking). The output from the editor is 11 XML files, one for each player detailing the positions the players should play in each mode, the style of play they should adopt in each mode and any actions the agents should take in each mode, e.g. where and when to pass, dribble and shoot.

7 Team Development

The HCIV development team consisted of 5 people from Linköpings Universitet in Sweden. About two man months were spent interfacing EASE and RoboCup and developing EASE player specifications by three under-graduate students, Tobias Wiren, Mikael Lönneberg and Pelle Nilsson. EASE was primarily developed by HCIV team leader Paul Scerri under the supervision of Nancy Reed.

Eight of the players that actually played in the 2000 World Cup games were untouched HCIII players and three were EASE players, primarily because the very high computational load of the negotiation algorithm prevented more HCIV players being used. All RoboCup specific aspects of the EASE players were developed from scratch except for server interface code which was adapted from HCIII. Most of the basic agent specification and agent runtime code of the HCIV was used without change from that used to produce simulated air-combat pilots.

The work was part of a larger project, funded by Saab Aerospace, NUTEK and CENIT, aimed at developing techniques for end-user specification of intelligent agents for simulation environments.

7.1 Relevant Webpages

See the Headless Chickens webpage at <http://www.ida.liu.se/~pausc/RC99/main.html> and the EASE webpage at <http://www.ida.liu.se/~pausc/Project.html>.

References

1. Rodney Brooks. Intelligence without reason. In *Proceedings 12th International Joint Conference on AI*, pages 569–595, Sydney, Australia, 1991.
2. Petru Eles, Krzysztof Kuchcinski, and Zebo Peng. *System Synthesis with VHDL*, chapter 5 "Optimization Heuristics". Kluwer Academic Publisher, December 1997.
3. Maja Mataric. Behavior-based systems: Main properties and implications. In *IEEE International Conference on Robotics and Automation, Workshop on Architectures for*, pages 46–54, Nice, France, May 1992.
4. Paolo Pirjanin. *Multiple objective action selection and behavior fusion voting*. PhD thesis, Dept. of Medical Informatics and Image Analysis, Aalborg university, 1998.
5. C. R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publishing, London, 1993.
6. Julio Rosenblatt and Charles Thorpe. combining multiple goals in a behavior based architecture. In *Proceedings of 1995 International Conference on Intelligent Robots and Systems (IROS)*, Pittsburg, PA, 1995.
7. Paul Scerri and Nancy E. Reed. Real-time control of intelligent agents. In *Technical Summaries of the Software Demonstration Sessions, Agents 2000*, pages 28–29, June 2000.
8. Paul Scerri, Nancy E. Reed, and Anders Törne. An approach to directing intelligent agents in real-time. In *Proceedings of the AAAI Spring Symposium on Agents with Adjustable Autonomy*, pages 114–115, 1999.
9. Paul Scerri, Johan Ydrén, and Nancy E. Reed. Layered specification of intelligent agents. In *Sixth Pacific Rim International Conference on Artificial Intelligence (PRICAI 2000)*, pages 565–575, August 2000.
10. S. Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 1996.