Hazard: A Framework Towards Connecting Artificial Intelligence and Robotics

Peter J. Andersson[1]

Department of Computer and Information Science, Linköping university, SE-58334 Linköping

Abstract. The gaming industry has started to look for solutions in the Artificial intelligence (AI) research community and work has begun with common standards for integration. At the same time, few robotic systems in development use already developed AI frameworks and technologies. In this article, we present the development and evaluation of the Hazard framework that has been used to rapidly create simulations for development of cognitive systems. Implementations includes for example a dialogue system that transparently can connect to either an Unmanned Aerial Vehicle (UAV) or a simulated counterpart. Hazard is found suitable for developing simulations supporting high-level AI development and we identify and propose a solution to the factors that make the framework unsuitable for lower level robotic specific tasks such as event/chronicle recognition.

1 Introduction

When developing or testing techniques for artificial intelligence, it is common to use a simulation. The simulation should as completely as possible simulate the environment that the AI will encounter when deployed live. Sometimes, it is the only environment in which the AI will be deployed (as is the case with computer games). As there exist many more types of environments than AI techniques, there is a need to reuse existing implementations of AI techniques with new environments. AI frameworks often come bundled with an environment to test the AI technique against and the environments are often not as easy to modify as a developer would want, nor is it easy to evaluate the framework in new environments without time-consuming development of middleware. In the case of robotics, there is an extended development period to create low-level control programs. The AI then developed is often tailored to the low-level control programs and it is hard, if at all possible to reuse.

The Robocup initiative [1] uses both actual robotic hardware and simulation in competition. Yet, there exists no common interface for using simulation league AIs with robotic league robots. This can mean that the simulation interface is unintuitive for actual robotics, or that AIs developed with the simulation are not usable with actual robots. In either case it is a problem worth investigating.

The WITAS Unmanned Aerial Vehicle project [2] uses several simulators in their research, both for hardware-in-the-loop simulation of the helicopter hardware and for development of dialogue interaction with an actual UAV. A middleware translating actions and events from WITAS protocol to other protocols would allow experimentation

J. Andersson1 P. (2005).
Hazard: A Framework Towards Connecting Artificial Intelligence and Robotics.
In *Proceedings of the 1st International Workshop on Multi-Agent Robotic Systems*, pages 171-176
DOI: 10.5220/0001193901710176
Copyright © SciTePress

with for example SOAR [3] as a high-level decision system. It would also allow the application of developed agent architecture to other environments and problems.

By creating a middleware framework that can mediate between low-level drivers and high-level decision system, we hope to be able to alleviate the problems for AI researchers presented above and inspire researchers in both artificial intelligence and robotics to reuse existing implementations. Robotics researchers can reuse AI techniques that exist and AI researchers can test their AI implementation in off-the-shelf simulators before building an expensive robotic system. It would also allow separate research groups to work with low-level control and high-level decision issues.

As a step towards proposing an interface and middleware for connecting AI and robotics, we have designed and implemented a framework for developing agents and environments based on action theory where we focus on the distinction between agent and environment. The design iterations and various implementations with the framework have allowed us to gain valuable experience in designing both interface and framework. The work is presented here together with an evaluation of the strengths, weaknesses and limitations.

2 Iterations of Design and Implementation

The strength of a design proposal may be measured in terms of its development history. Designs that have undergone several iterations under different conditions are much more mature than designs without practical grounding. The Hazard framework was derived from the Haphazard Role-Playing Game project and refined through several iterations that are presented below.

2.1 Haphazard - An Online Role-Playing Game

The Haphazard Role-Playing Game [4] started as an open-source project for creating a simple online role-playing game with focus on AI implementations. It was from this game project that the first version of Hazard was extracted. The framework was then redesigned and Haphazard was reimplemented to use the new framework. Haphazard has the most complex environment of all implementations up to date, but only rudimentary AI. The Haphazard project was the most prominent project when designing the framework for environments.

Environment
: The environment in Haphazard is a grid-based model using tiles for visualisation. Objects in the world are either static or dynamic. Static objects include houses, trees and other scenery. Dynamic objects are the objects that can be manipulated by the player.

Agents
: Agents are either controlled by AI or by a player through a graphical user interface with the actions move, equip, carry, drop or use.

2.2 Simulator for Evaluating the Subsumption Architecture

An implementation of the subsumption architecture [5] was introduced to the agent part of the framework as part of the work towards evaluation the subsumption architecture for use in obstacle avoidance systems for cars [6]. This implementation allowed us to evaluate the agent part of the framework and enhance it. The subsumption architecture was integrated with an editor which allowed the user to change the subsumption network during simulation runs to experiment with new behaviours.

Environment
: The subsumption agent was used in two environments. The architecture was developed and tested in the Haphazard environment, but mainly used in a traffic simulator. The traffic simulator used a straight road without any static obstacles.

Agents
: The user controlled a car with acceleration and turning, the agent modified the user's input with a subsumption network before it was actuated to avoid obstacles.

2.3 Simulator for Dialogue System Development Support

Within the WITAS Unmanned Aerial Vehicle (UAV) project [2], the Hazard framework was used in implementing a simulator that supports development of a dialogue system for command and control of one or more UAVs.

Environment
: The environment consists of a three-dimensional map containing roads and buildings. Buildings are visualized as polygons and have a height. All buildings also have a name, color, material and one or more windows which can be observed by sensors.

Agents
: There exist two types of agents:

  UAV
  : The UAV agent consists of a socket interface which can receive commands from both a dialogue developed within the WITAS project and the COMETS Multi-Level Executive developed within the COMETS project [7]. The simulator executes these commands and report their progress. The agent has a camera sensor and can detect cars and buildings. The UAV can take off, land, hover, fly around, follow cars and follow roads to accomplish it's mission. It is an interactive agent that can build plans and follow the commands of a user.

  Car
  : Cars drive along roads with a set speed.

The simulation is transparent and can be fully or partly substituted by a connection to a real world UAV. The visualization is 2D, but current work is extending both camera view and world view to three dimensions.

3 Evaluation

The evaluation of the framework is based on our own experience in developing simulations. It is focused on development and how suitable the framework is for use as middleware and as a framework for development of high-level and low-level AI for robotic applications.

3.1 Strengths

Rapid development
: The different implementations have shown that the framework rapidly gives a working system. The only comparison that has been done on development time is with the replacement of the Simulator for Dialogue System Development Support. The implementation using the Hazard framework cut the development time radically.

Scalability
: The framework is very scalable in the form of developing new agents or objects for an environment. It has not been tested how scalable it is in regard to the number of existing objects/agents in an environment.

3.2 Weaknesses

Agents are tightly linked to the environment
: Since the framework was extracted from an integrated system, the agents are tightly linked to the environment and can have complete knowledge of the world without using sensors. The agent module should be completely separate from the environment.

Agent to agent interaction
: Since the design does not distinguish between success/fail messages for an action and sensor data, it is hard to develop agent to agent interactions. A solution for this problem could be to remove the success/fail notion from the environment side of the action and let the agent side sensor interpreter decide when an action has been successful or failed. This solution would also allow research into event/chronicle recognition.

New developers
: The system is well documented, but lacks examples and tutorials. This makes it harder for new developers to use the framework.

3.3 Limitations

Mid-level functionality
: Since the framework only supports high-level actions and is confusing on the part of sensor data, mid-level functionality is not intuitively supported. By mid-level functionality is meant functionality such as event/chronicle recognition, symbol grounding and similar robotic tasks. This is a disadvantage if the system is used for developing AI techniques for robotic systems since a developer can easily “cheat” with constructed events instead of having to identify them from sensor data.

Pre-defined set of actions

Since the actions are pre-defined in the environment, both with regard to execution and evaluation of the execution (success/fail), an agent cannot learn new actions or interpret the result in new ways. Also, since the actions can be of different level of abstraction, it is hard to combine actions concurrently.

4 Related Work

Currently, the International Game Developers Association (IGDA) is pushing towards AI standard interfaces for computer games and is in the process of publishing the first drafts of a proposed standard. These standards are geared towards enabling game AI developers to reuse existing AI middleware and to concentrate on higher level AI tasks. IGDA is working on interfaces for common AI tasks and has currently working groups on interface standards for world interfacing, path planning, steering, finite state machines, rule-based systems and goal-oriented action planning. The work presented here is closest to the work on world interfacing, but since the draft was not available at the time of writing, it was impossible to compare.

The Testbed for Integrating and Evaluating Learning Techniques (TIELT) [8] is a free software tool that can be used to integrate AI systems with (e.g., real-time) gaming simulators, and to evaluate how well those systems learn on selected simulation tasks. TIELT is used as a configurable middleware between gaming simulators and learning systems and can probably be used as a middleware between general environments and agent frameworks with some modification. The middleware philosophy of TIELT differs from our implementation, TIELT sits as a black box between environment and agent while Hazard is only meant as a transparent interface without any real functionality, except if the framework is used on either side. The black box functionality would hide event/chronicle recognition, symbol grounding, etc. . . in a robotic system. This means that special care has to be taken if the system is to be used with actual robotics.

The System for Parallel Agent Discrete Event Simulator (SPADES) [9] is a simulation framework with approximately the same concept as the Hazard framework with regards to the separation between agents and environments. It focuses on repeatability of simulations and uses a concept of Software-in-the-Loop. SPADES is a discrete event simulator and uses a sense-think-act loop for the agents which limits its deliberation time to the time between receiving events until it has decided what to do. This limitation is minimized by allowing agents to tell the simulation that it wants to receive a *time notification* which works as a sense event. Our framework on the other hand sacrifices repeatability and agent timesharing to obtain a continuous, asynchronous time model which is more inline with robotic architectures than a discrete model.

There is also research on modules and frameworks that can be used in conjunction with a set of interfaces for cognitive systems, in particular DyKnow [10].

5 Conclusions and Future Work

The iterative development of framework and interfaces has enabled us to gain valuable experience in designing interfaces that are adequate for both robotic systems and simulated environments without sacrificing detail or ease of use. Our goal is to develop an architecture for connecting AI and robotics with the following characteristics:

- Rapid environment/agent development
- Able to reuse both existing agents and environments
- Capable of acting as middleware between existing frameworks
- Usable in research of both simulations and actual robotic systems

Hazard is a mature system which has undergone several design iterations. It allows rapid development and reuse of agents and environments. It contains interfaces between agent and environment and can be used as middleware between existing frameworks. But Hazard has been found unsuitable for development of AI or simulations for actual robotic systems due to its inherent limitation in event recognition and actuator control. To be usable in a robotic AI implementation, the interfaces need to be layered to allow both for high-level AI frameworks and middle-level event and chronical recognition on the agent side.

Currently, work has been started on a new generation of interfaces and framework. This work is called CAIRo (Connecting AI to Robotics) and addresses the issues found with the development of Hazard.

References

1. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E.: RoboCup: The robot world cup initiative. In Johnson, W.L., Hayes-Roth, B., eds.: Proceedings of the First International Conference on Autonomous Agents (Agents'97), New York, ACM Press (1997) 340–347
2. Doherty, P., Granlund, G., Krzysztof, G., Kuchcinski, K., Sandewall, E., Nordberg, K., Skarman, E., Wiklund, J.: The witas unmanned aerial vehicle project. In: ECAI-00, Berlin, Germany (2000)
3. Laird, J.E., Newell, A., Rosenbloom, P.S.: Soar: An architecture for general intelligence. Artificial Intelligence **33** (1987) 1–64
4. Andersson, P.J., Beskid, L.C.: The haphazard game project. http://haphazard.sf.net (2003)
5. Brooks, R.A.: A robust layered control system for a mobile robot. Memo 864, MIT AI Lab (1985)
6. Woltjer, R., McGee, K.: Subsumption architecture as a framework to address human machine function allocation. Presented at SimSafe, http://130.243.99.7/pph/pph0220/simsafe/dok/simsafe05.pdf (2004)
7. Ollero, A., Hommel, G., Gancet, J., Gutierrez, L.G., Viegas, D., Forssén, P.E., González, M.: Comets: A multiple heterogeneous uav system. In: Proceedings of the 2004 IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR 2004), Bonn (Germany) (2004)
8. Aha, D., Molineaux, M.: Integrating learning in interactive gaming simulators. In Fu, D., Orkin, J., eds.: Challenges of Game AI: Proceedings of the AAAI'04 Workshop (Technical Report WS-04-04), San Jose, CA, AAAI Press (2004)
9. Riley, P.F., Riley, G.F.: Spades - a distributed agent simulation environment with software-in-the-loop execution. In Chick, S., Sanchez, P.J., Ferrin, D., Morrice, D., eds.: Proceedings of the 2003 Winter Simulation Conference. (2003) 817–825
10. Heinz, F., Doherty, P.: Dyknow: An approach to middleware for knowledge processing. Journal of Intelligent and Fuzzy Systems **15** (2004)