M.Sc. in Interactive Entertainment Technology

CS7032: AI for IET

Creating a simple robocode agent

Ben Thompson
Student No.
16337900

October 11, 2016

Abstract

The process of creating a good robocode agent requires the analysis of its environment and functional capabilities. Both P.E.A.S. and the environment properties can be viewed in the tables below.These methods of classifying agent and environment properties can be used to justify the choice of abstract architecture used.

Agent and Environment properties:

Environment properties	
Property	Description
Partially observable	Robocode agents do not receive full information on their environment. They are limited to their radar which has a scan distance of 1200 pixels with limited turn rate and collision detection of both the robot and robots bullets.
Stochastic	Robocode agents deal with a high level of uncertainty in their environment and thereby the current state and action performed by the agent do not determine the next state.
Episodic and Sequential	Robocode agents rely on an assessment of both current state and past states of the environment to decide on the next action.
Dynamic	The environment changes independently of the agent due to the presence of other agents.
Continuous	The ability to turn, move and fire in 360 degrees make for infinite possibilities.
Multi-agent	Robocode at its core is about pitting two or more agents against one another in an arena type environment and thereby should always involve multiple agents.

P.E.A.S.			
Performance measure	Environment	Actuators	Sensors
• Positive performance measures – Hit target – Not being hit – Scanning enemy – Ramming enemy • Negative performance measures – being hit – not hitting a target – being rammed	• 2D pixel world (Rectangle or Square) • Enemy and/or Friendly agents	• Firing • movement (turning and moving forward and back)	• Scan (radar) • Collision detection (robot and robots fired bullets) • being hit

Abstract Architecture

Given that the robocode environment has both sequential and episodic aspects a hybrid architecture with both reactive and deliberative elements could be seen as optimal. In the java files submitted, the layout of the robot resembles that of a TOURINGMACHINE in that it is divided into three layers. The modelling layer, the planning layer and the reactive layer.

The modelling layer is presented as a static nested class of the robot. All aspects of the environment are stored there statically in order to create a symbolic representation of the world. The planing layer is contained in the main robot class. In this case a state based architecture is used to change the set of both default actions in the run method and reactive actions triggered by events. The reactive layer is subject to change as just stated by change in state caused in the planing layer.

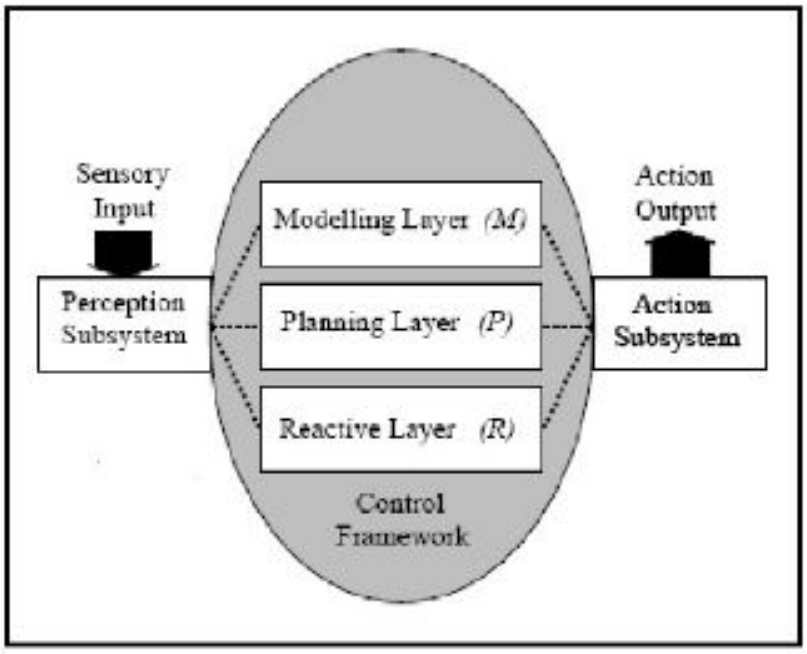

Figure 1: TOURINGMACHINE architecture.

The strategy of the robot is merely going from the scan state doing a 360 degree scan and sending the number of enemies scanned to the model representation. Then in the planning layer if the robot has done a scan and depending on the amount of enemies scanned it will change to an appropriate state. The planing layer waits ten runs then does another scan and repeats the process.All of these combine to make the control framework.

TOURINGMACHINES are implemented using horizontal layering, this means that each layer itself acts like an agent. Given this is the case both the planing and reactive layers will described accordingly.

Reactive layer:
Purely reactive agents can be modelled under the assumption

$$action : S \rightarrow A$$

The standard 4 tuple architecture remains so for reactive elements

$$Arch =< S, A, action, env >$$

One example of a purely reactive architecture used would be the On-RobotScanned used in the scan state.$RobotScanned \in S$ In the case of the corresponding action, $CountUniqueRobot \in A$. If a robot is scanned and has a unique name then the number of robots scanned is incremented, $RobotScanned \rightarrow CountUniqueRobot$.

Planing Layer (State based architecture):

$$Arch =< S, A, P, I, action, env, see, next >$$

$$see : S \rightarrow P$$

$$action : I \rightarrow A$$

$$next : I \times P \rightarrow I$$

In the case of the planing Layer. The robot uses the symbolic representation in the world to change behaviour sets (states). The strategy is merely to scan the environment and decide on a state based on the amount of scanned robots. This would be represented as such...
Facts perceived by the agent represented by P

$$ScanedMoreThan3Robots \in P$$

The decision function becomes for this

$$action : ScannedMoreThan3Robots \rightarrow ChangeToSuitableState$$

This is of course linked to the environment states via the perception function

$$see : S \rightarrow P$$

Given the robot is a hybrid architecture, capturing the design comes with its difficulties.Hybrid architectures themselves are often described to be very specific and application dependent, unsupported by formal theories and lacking general design guiding methodologies.

References

M.Wooldridge:An introduction to Multi-agent Systems chapter 5 (reactive, hybrid)

Brooks, R. (1986). A robust layered control system for a mobile robot. IEEE journal on robotics and automation, 2(1):1423

Andrei Marinescu: IET Lectures 1-3 https://www.cs.tcd.ie/ amarines/teaching/cs7032/