Design issue of a multi-agent system called Ruby CUP

This is an idea of making a automatic football system using multi-agent system concepts and techniques. In order to put it into the reality I mention some design issue in this article.

Generic Features

In this system there is a football playground with 2 doors, two teams of players(team A and team B , each contains n players, n is, for example, 3) and one ball. Players aim to shut the ball at other team's door.

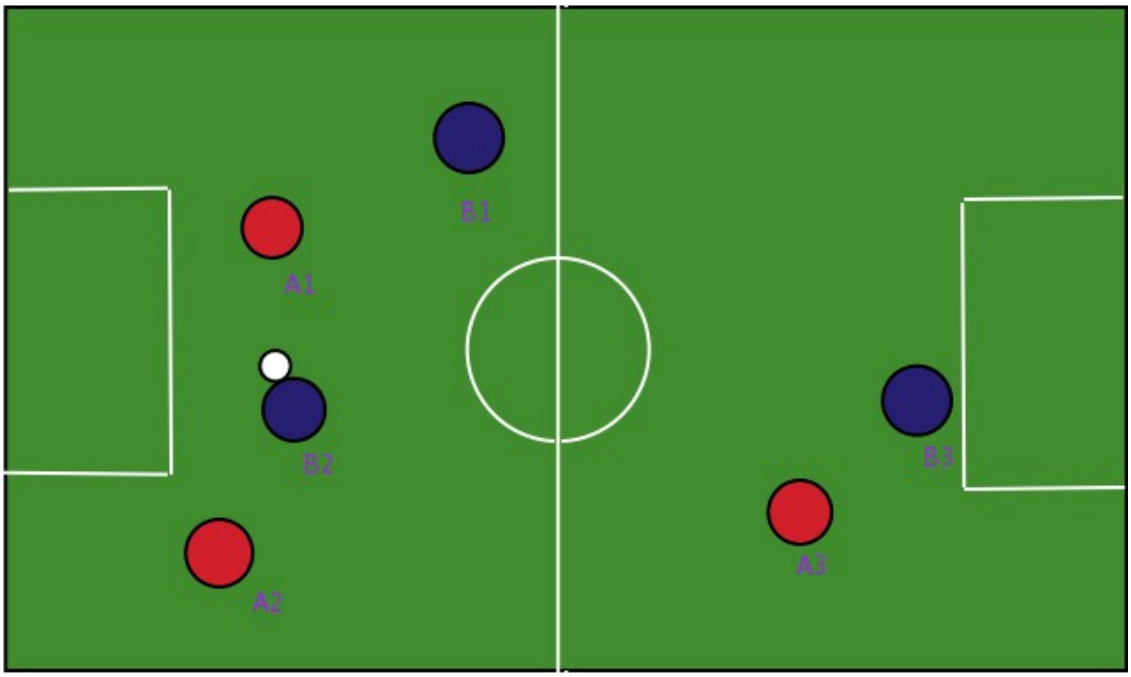

Figure 1. Playground

The playground is a closed field, players run and act on it during a match, When player or ball touched the border of the playground then then it's mirrored back, so it cannot go out of the playground. Playground has fixed width and height that can be know for player. Whenever the ball goes into one of the two doors , the door's owner's rival got one score which will be accumulated until the match ends. The team with higher score wins, the other lost.

Player can hold the ball and take it with him as he touch the ball. When two players hit, they bounce away from each other, the ball they take is bounced away too and become free with an initial velocity and direction.

Players can change their velocity and directions in a range (velocity in [0, VMAX], direction in [0, 360]) to perform their actions. That is the only point intelligence can do on players. Player can communicate with his teammates to collaboratively making strategies and tactics, like passing ball to others.

When the ball is freed, it keeps going with its direction and velocity that keeps reducing with coefficient (V[n] = V[n-1] * C, C is a coefficient such as 0.99) until its velocity goes near 0 or it was hold by some players.

Two teams and observers have different logical coordinates as depicted in the figure 2

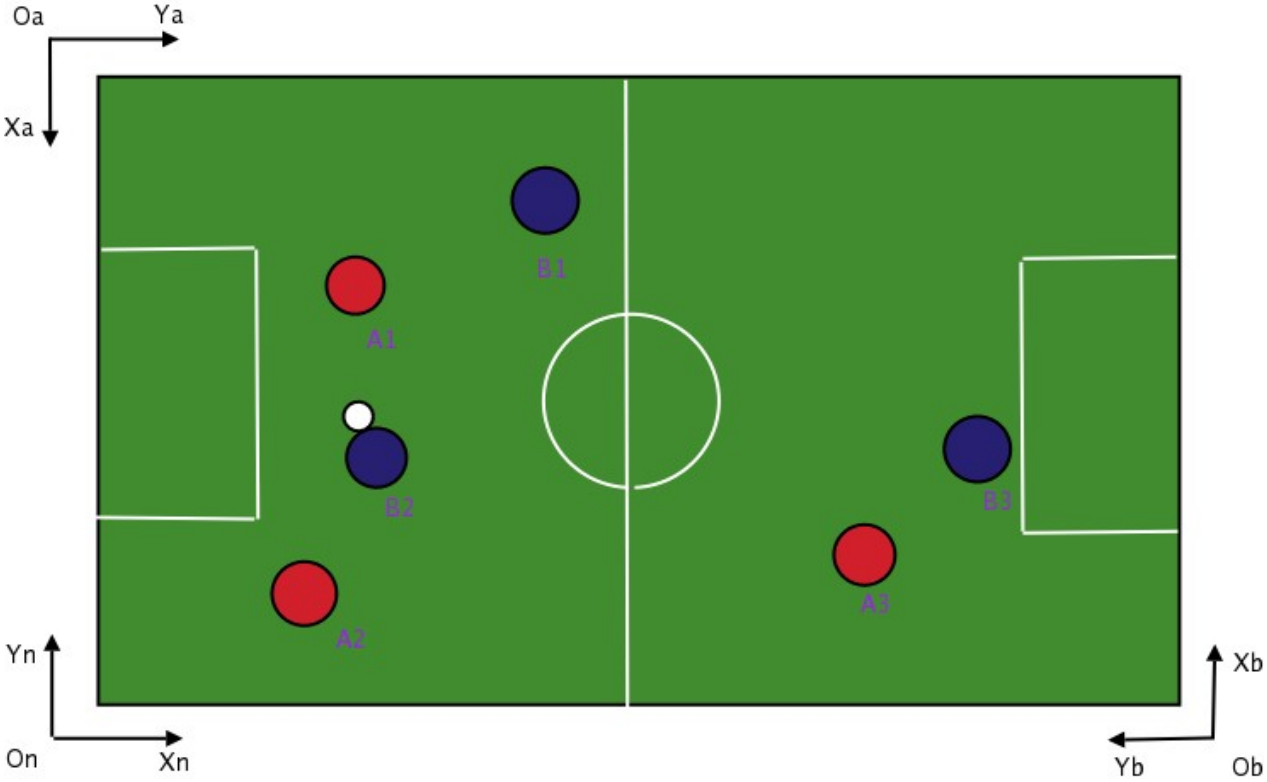

Figure 2. Different coordinates

Xa-Oa-Ya is the coordinates of team A, Xb-Ob-Yb is that of team B, while Xn-On-Yn is that of Observers. The reason of distinguishing coordinates is that teams should have unique view of the whole playground (that is, their door is at the bottom of the playground while rival's door is at the top) and observers' view should be in the neural and usual view. So there have to be coordination transforms.

Architecture of MAS System

I plan to implement above topics in Multi Agent System utilities. In the logical view, agents are autonomous, intelligent and distributed objects. There is a unique AP(Agent Platform) for one MAS application, one AP hold one or more AC(Agent Container) one of which is the root AC. An AC resides in a single machine. AP plus AC, as the container of agents, provide environment for running agents. Agents are all actors in a MAS application, it has states and is able to observe the environment, think over observations and perform actions on environment to change the state of environments. Users make agents of their own to play in this platform.

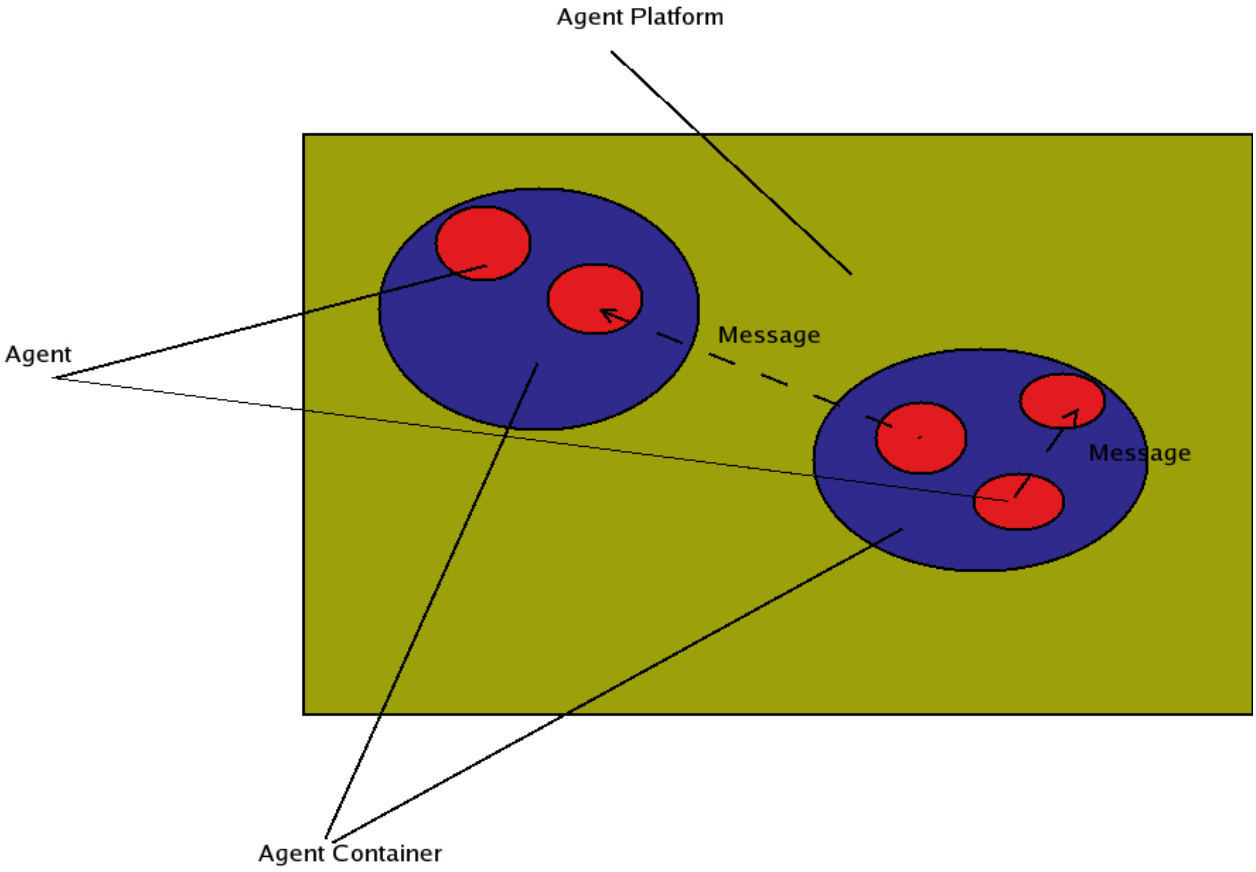

Figure 3. Multi Agent System

Agents communicate through single directional Messages. Message have multiple fields to expresses things like sender's modes of mind and message content.

An example of message in XML format is

```
<message xmlns="mas:message" performative="REQUEST">
  <sender><aid name="Mike"/></sender>
  <receivers>
      <aid name = "Jerry"/>
      <aid name = "Tom"/>
  </receivers>
  <language>English</language>
  <ontology>ask.for.help</ontology>
  <content>Give me an apple</content>
  <reply-by>344243.23234.112@localhost:7812</reply-by>
  <reply-with>2046-08-17 14:26:00 </reply-with>
</message>
```

Agent catagory

Actors are agent in this system, They are:

Match: The agent entity to hold life cycle of a match, broadcast current status, receive player state changes , maintain scores and determine who's the winner.

Player: The agent on behalf of a football player who can run, hold the ball, passing ball to

teammates and shut at the door. Users may custom their own player agents to make own actions, multiple brains of a player can be switched.

Observer: User interface agent to show the detail of a Match , there may be multiple Observer that are registered into the broadcast list of a Match agent to receive match status. I have made a primitive observer agent using SDL ruby binding

Coach(Optional): Team instructor to hold a strategic view over the whole team, He sends instructions(or more precisely suggestions) to his team players. it is up to the user to determine if a coach agent is needed.

Basic reactive behavior

The fundamental behavior of agent is to perceive , think and behave once and more. The perception at the nth loop has a input Cn. Thinking is a function of a sequence of perceptions , think = function(C1, C2, ..., Cn) and action is the outcome of thinking.

We call the behavior reactive when its actions depend only on the current perception Cn instead of any historical perceptions, a basic Ruby CUP's behavior can be considered as a reactive behavior. In this case, the state of player can be determined with the current ball status. As for a player

If he is holding the ball, his state is "hold", Else if one of his teammate is holding the ball, his state is "assist", which means to help the ball holder shut at the door. Else if his team's rivals hold the ball, his state is "defend", it means to defend against rivals' attack. Else, the only case is that the ball is hold by neither of the two teams, his state is "idle" which means to contend the ball if possible.

As a reactive agent, its behavior is simple:

In case of "hold" state, the player shuts at the door if he is adjacent enough to the door , or else he tries to move to that position.

In case of "defend" state , the player goes to contend the ball as the rival holder enters his half ground.

In case of "idle" state the player goes to contend the ball if he is the nearest to the ball among his teams.

In case of "assist" the player just goes randomly. to target is assigned.

Now the reactive agents are runnable, the match using that intelligence is good look except for some stupidness.

Add collaborations: passing of ball

The biggest problem of reactive behaviors is that there is no coordination, players fight for himself. but why there are teams? Why don't set up 6 doors with 6 players fight against one another? Players should know that they have to collaborate in order to win.

The collaborating is transporting balls, it uses a well known pattern in the multi-agent communications: the Contract Net. a basic procedure runs like this:

S1: An initiator has projects, he sends calling for proposal messages to many responders

S2: Those responders bide with a prices

S3: The initiator evaluates those bides, chooses the best applicable responders and gives him the offer.

That's it.

This procedure can be mapped into the passing of balls.

S1: When ball holders finds no way of penetrating rivals' line of defense . He calls for a ball passing(by sending a CFP message) to his teammates.

S2: His teammates evaluate their position before giving out a number showing how it would be easy to pass the ball.

S3: The ball holder passes ball to the most easily reachable teammate.

Another pattern of ball passing is:

S1: when an assistant thinks that he is more suitable to take ball than the ball holder, then he sends a message to ball holder request for a ball passing.

S2: The ball is passed towards the assistant if the ball holder agrees.

After the two Wisdoms are spawned, they are placed into the same playground for player kill. The score rate is 1: 10, So you can guess who's who.

In 2 weeks time the basic football agent system should be running, all are written in ruby language(It is also possible to make a language-neutral system in the future because all messages can be transported using HTTP/TCP protocol instead of using the Druby implementation), all agents can be hold in one or more containers, player agents have basic intelligence and a more intelligent agent will be made out. More amendments can be attached according to working schedules.