Cooperative Architecture for Multi-Agent Systems in Robotic Soccer (CAMASS)

Julian Angel, Fernando De la Rosa, Germán Bravo
Systems and Computer Engineering Department
Universidad de los Andes
Bogotá, Colombia
quetzacoalt@gmail.com, {fde, gbravo}@uniandes.edu.co

Abstract—In order to create a cooperative architecture for multi-agent systems in robotic soccer, it is necessary to model and understand the human soccer concepts to use and express them computationally. This work presents the main features for a multi-agent cooperative architecture based on human soccer concepts. The architecture can be applied to different categories of RoboCup. To test its performance, an initial development was accomplished for the 2D RoboCup category, obtaining a system able to execute synchronized simple tactics.

Keywords: robot software architecture, teamwork, multi-agent system, robotic soccer

I. INTRODUCTION

Soccer is a highly dynamic sport where each second taken for a player to think for the next best action can affect the plan’s validity: in several circumstances, it may be better to do a common action than to do nothing at all. In addition, each player makes his decisions autonomously and based on a partial game state. When players play in a selfish and unsynchronized way, we only observe a lot of players following the ball.

In this work, we present a Cooperative Architecture for Multi-Agent Systems in Robotic Soccer (CAMASS) based on human soccer concepts. CAMASS consists of five decision levels corresponding to different abstraction levels, organized as a hierarchy. These decision levels in decreasing abstraction level are: strategy, complex tactic, simple tactic, play, and skill. The cooperation and synchronization aspects of actions are supported by the three upper levels (strategy, complex and simple tactic) and the communication between the agents. To facilitate the loading of information required by CAMASS to support the decision process, a description language DLCAMASS, was designed to specify strategies, complex and simple tactics, as well as training tactics from files. This language allows the definition of test case scenarios for each level, without having to change the code of the system, reducing the time needed to change the set of strategies for games.

This paper is organized as follows. Section II discusses various architectures and methods of cooperation used in robotic soccer. The description and formalization of the CAMASS’s decision levels are presented in section III. Section IV shows an implementation for the RoboCup 2D simulation league. Results obtained in the processes of creation and testing are shown in section V. Finally, section VI presents some conclusions and proposes some outcomes for future work.

II. RELATED WORKS

Being the focus of this work the use of human soccer concepts to propose a cooperative architecture in robotic soccer, the analysis to related works is restricted to robotic soccer. One of the most prominent works used as a basis for other teams in the RoboCup 2D category was developed by P. Stone [1], [2]. This work proposes that an agent must keep three types of states: the world state with the information of the real-world which is updated from the agent’s sensors, the locker-room agreement, created privately by the team, keeping the team structure and the communication protocols between agents, and the internal state that keeps the agent’s internal data.

Other works for other categories include the team CMDragons03 [3], designed and implemented as a cooperative architecture for the RoboCup small category defining three decision levels: (a) play, defining a fixed plan with a set of pre-conditions, post-conditions, actions and N roles (where N is the number of players in the team), (b) tactic, encapsulating the behavior of a single robot, and (c) skill, defining the control policies that enable the robot to execute complex behaviors. MRCC is a proposal of a multi-layer cooperative architecture [4], [5], [6] where the upper layers deal with general aspects of the problem and restrict the functionality of the lower layers. This architecture is composed of four layers: Strategy, Formation, Cooperative Action and Agent. CS Freiburg [7] is a team that participated in the RoboCup middle category, where the actions performed by a role are subject to the location of the team’s formation. TABA [8] is a cooperative architecture designed for the RoboCup four legs category where a leader agent is charged of selecting the plan that the whole team will execute. Kontes and Lagoudakis [9] propose the use of Petri nets to describe the actions related to a role. Each of these proposals uses the soccer concepts differently, depending on its own needs. A work which highlights the use of human soccer concepts was conducted by Dylla et al. [10],[11], but this work only focuses on defining a model of the human knowledge without worrying about the implementation.
The main features observed in these works can be summarized in three categories: computational concepts, horizon time of plans, and human concepts.

The computational concepts refer to the use of concepts and techniques related to computational systems. The main features observed are: (a) the use of multi-agent systems (MAS) paradigm, (b) the use of roles for the assignment of tasks and responsibilities, (c) the use of multi-layer architecture, allowing the generation of plans by decomposition, (d) the portability, understood as the capacity of the architecture to be applied to different categories and robotics soccer platforms, and (e) the use of computational tools to specify the initial information and/or to facilitate the testing of the system.

The horizon time of plans refers to the time required for a plan to reach completion. Horizons times between less than 1 second and 300 seconds, are considered and summarized in Table I.

**TABLE I.** HORIZON TIME OF PLANS

<table>
<thead>
<tr>
<th>Horizon time of plan</th>
<th>Average time to plan completion (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instant – Skill</td>
<td>0 ≤ t ≤ 1</td>
</tr>
<tr>
<td>Short – Play</td>
<td>1 ≤ t ≤ 2</td>
</tr>
<tr>
<td>Medium – Simple Tactic</td>
<td>2 ≤ t ≤ 12</td>
</tr>
<tr>
<td>Long – Complex Tactic</td>
<td>12 ≤ t ≤ 30</td>
</tr>
<tr>
<td>Very long – Strategy</td>
<td>30 ≤ t ≤ 300</td>
</tr>
</tbody>
</table>

Human concepts refer to the use of concepts used in human soccer in the decision process and or in the architecture. The considered concepts are: strategy, tactic, play, skill, role and formation.

The comparison of studied works appears in tables II, III and IV, where it is important to highlight the generalized use of MAS. In addition, all studies use instant and short-time horizons, and the human concept of formation. The use of both, instant and short plans, allows the players to respond to the natural dynamism in soccer. Nevertheless, every work adapts the meaning of the concepts to suit its own needs and often these definitions differ from the common meaning, making difficult the understanding of the proposed systems. There is also evidence that not all works were thought to be used in different categories, or they do not usually plan with long horizon time, or they have limitations to reuse the defined plays under slightly different conditions.

**TABLE II.** USE OF COMPUTATIONAL CONCEPTS.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SMA</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Roles</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Architecture</td>
<td>±</td>
<td>✓</td>
<td>±</td>
<td>NA</td>
<td>✓</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Portability</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>±</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Language</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>±</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

**III. CAMASS**

The architecture of CAMASS results from the use of soccer human concepts. The main characteristic of this architecture is the definition of multiple decision levels: strategy, complex tactics, simple tactics, play and skill. Strategy is the higher abstraction level, concerned by global aspects of the game like the score, remaining playing time and attitude of the opposing team. On the other hand, skill is the lowest level of abstraction level, determined only by the play being executed.

CAMASS foundations are the soccer human concepts described in [14], described below in ascending complexity order, are:

*Primitive:* Primitives are the actions that a robotic platform can execute. For example, an average Robotics platform in RoboCup small category has the following primitives: move (X, Y), rotate (final bearing), move - rotate (rotate while moving), activate trigger and activate dribbling.

*Skill:* The concept of skill is taken from [3], allowing the use of an architecture in different robotic platforms (portability). A skill is defined as the sequence of primitive actions to be performed in a robotic platform to execute a play. For example, for a small category platform to get a ball, it must perform the following primitive actions: move (X, Y, final bearing), and turn on the dribbling mechanism. Skill is the lower abstraction decision level, concerned only to the choice of the best sequence of primitives to perform a specific play. If there is no such sequence, this level notifies the upper level (play), to let it to make a decision.

*Play:* A play is parameterized action carried out individually by a player. Considered plays are: shoot, dribble, move, mark and intercept. It is important to note that in this work the pass is not considered a play, because is seen as the combination of two actions, shoot and intercept, performed by two different players. The play decision level is responsible of choosing the appropriate skills to perform the desired play. This choice occurs every time the world is updated, or a play is finished, or the execution of a new play is commanded by the upper level. When the play finishes or cannot be performed, the play level notifies the upper level to let it to make a decision.
**Tactic**: A tactic is a sequence of plays for each player in the field. In a tactic, every player has his own sequence of plays; the composition of all sequences for all players constitutes the tactic been played by the team. It is expected to see all the players to move and cooperate as a whole. A team may have as many tactics as it wishes. The length of these sequences must be chosen carefully in order to guarantee their successfullness; too long sequences have a low probability to finish completely. The strategy to solve this problem is based on how play human soccer players: they are aware of the tactic been played, but also know some variants and are also capable to move to another tactic. These considerations lead to two kinds of tactics: Simple and Complex tactics.

**Simple Tactic** is a sequence of two to six parameterized plays (medium term plans). CAMASS considers three types of simple tactics: 1) Normal, st, which correspond to the normal flow of the tactic. 2) Transition, tt, which allow passing from a complex tactic to another. 3) Panic, pt, used when there is no possible tactic. The Simple Tactic decision level, running at every player, is responsible of choosing his next play, based on the knowledge of the Simple Tactic been followed by the team. The choice of appropriate play is based on the location of some players and/or the ball location. The decision is taken every time the lower level notify that the current play has finished, or when the play level notifies the failure of the current play, or when the upper level commands the execution of a new tactic. In case of failure of the current Simple Tactic, this level notifies the upper level to let it to make a decision. Fig. 1 shows an example of a simple tactic (withdrawal), which must be triggered when the team lost the ball and commands to all players to go to a defensive position in the shortest time. When players reach their position, it would be possible to pass to another different tactic, as pressing, for example.

**Complex Tactic** (ct) is a directed graph, where nodes are Simple Tactics and arcs are the conditions to move from one node to another (long-term plans). There are two types of complex tactics: offensive, when the team has the ball control, and defensive, when the opposing team has possession of the ball. A schematic example of a complex tactic is shown in Fig. 2. The Complex Tactic decision level contains several Simple Tactics, defining the ideal flow of the complex tactic and some variants to be used when the current simple tactic fails. It is responsible of choosing the appropriate Simple Tactic, based on the ball possession, the success of tactic been played, the current distribution of all players (including the adversary) and the general state of game. It has also the responsibility of log the tactics used, their frequency and success, in order to improve the tactics for other matches.

![Figure 1. Example of a simple tactic (st): withdrawal.](image1)

![Figure 2. Complex Tactic (CT) shown possible flows of Simple Tactics (ST) and some Transition Tactics (TT).](image2)

**Formation**: A formation defines the location of every team player in the field, and in this way defines also the desired roles for every player.

**Role**: A role encapsulates all favorite skills and plays of a player, similar to positions and roles of players in human soccer. The roles considered in this paper include: goalkeeper, defender, sweeper, full-back, center midfielder, side midfielder, center-back, center-forward, and striker [14].

**Strategy**: A strategy is composed of defensive and offensive Complex Tactics, joined by Simple Transition Tactics (very-long term plan). A strategy is structured as a directed graph (Fig. 3) where the nodes are Tactics, Simple as well as Complex, and arcs represent sequences of Tactics, with the following integrity rules: (1) Simple Tactics are Transition Tactics, (2) there cannot occur a sequence of two Tactics of the same type, (3) all Tactics has a link to Panic Tactic.

![Figure 3. Strategy level shown defensive and offensive Complex Tactics linked by Transition Tactics and the Panic Tactic (not all links to Panic Tactic are shown).](image3)

The definition of a strategy takes into account the concept of role, based on the intended formation to use during a match. For example, to obtain a formation 4-4-2, the set of necessary roles are: two center-backs, left side and right side defenders, two central midfielders, left and right midfielders, two forwards and a goalkeeper. It is natural to think that the set of tactics (simple or complex) must be chosen carefully, depending of roles desired for a given match. Being the upper decision level in the platform, it must consider the state of the game, as well as the objectives for the match, like for example, win the game, tie or not lose for more than x goals. Some environment
variables to take into account are: game period (first, second or extra time), goals scored by teams, yellow and red cards, characteristics of the opposing team (training, strategy, performance, …).

IV. IMPLEMENTATION

Being CAMASS conceived to use human soccer concepts, its first implementation is in the RoboCup 2D simulation category, which has all the characteristics of a human soccer match: a coach, eleven players, broadcast communication and limited listening skills for each player. The base of CAMASS implementation is the released WrightEagle’s 2D code [12], [13], which allows the connection, reception and sending of messages to a server, follows the category rules, maintains and updates the world that each player can see. The main changes to this code concern how the agents make and communicate decisions to their teammates. To make these changes, the use of activity diagrams was valuable. The decision-making process in each level of architecture (Table V) occurs in the following way:

- The coach can communicate with players every 30 seconds only, and having all the information of the match without noise, he is responsible for deciding the strategy that the team must follow.
- The choice of Complex and Simple tactics may be made by the coach, the team captain or the player who has possession of the ball. In case of conflict, the defined priority comes to the coach, the team captain or the player who has possession of the ball.
- Each player is responsible, by his own, of verifying the necessary conditions to continue with a given play.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Decision level</th>
<th>Responsibility</th>
<th>Horizon time of plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAS</td>
<td>Strategy</td>
<td>Select transition tactic to other complex tactic</td>
<td>Very long</td>
</tr>
<tr>
<td>Tactic</td>
<td>Complex</td>
<td>Select Simple tactic</td>
<td>Long</td>
</tr>
<tr>
<td></td>
<td>Simple</td>
<td>Select Play</td>
<td>Medium</td>
</tr>
<tr>
<td>Agent</td>
<td>Play</td>
<td>Select Skill</td>
<td>Short</td>
</tr>
<tr>
<td></td>
<td>Skill</td>
<td>Use Skill</td>
<td>Instant</td>
</tr>
<tr>
<td>Platform</td>
<td>NA</td>
<td>Execution</td>
<td>NA</td>
</tr>
</tbody>
</table>

Each player must know the strategies and tactics being used during the match. In CAMASS, this information is parameterized, and it is loaded from files at the beginning of the match, allowing the modification of strategies or tactics from match to match, without any further code changes. To facilitate the description of team strategies and tactics (simple and complex), CAMASS proposes a XML-based description language called DLCAMASS, and its corresponding parser were built. In this first version of CAMASS, the following plays are considered:

- **Zone marking (zone):** the player goes to its default zone and tries to intercept the ball. A possible improvement is to take into account the location of opponents and try to stay at the same distance of every opponent. Doing so, it is expected that the probability of intercepting the ball is increased.
  - **Player marking:** the player marks a specific opponent, trying to locate himself in a position allowing the interception of the ball when the opponent is trying to receive a pass.
  - **Dribble (x, y):** the player moves from its current position to another trying to avoid the opponents, and conserving the ball. Currently, the player just goes in straight line to destination.
  - **Shoot (x, y):** the player tries to shoot to a specific zone, trying to maximize the time that opponents would take to intercept the ball. Current implementation does not take account of the opponents.
  - **Intercept:** while in one zone, the player tries to gain control of the ball.
  - **Unmark:** the player looks for a location where he can receive the ball.
  - **GoTo (x, y):** the player, not having the ball, moves from its current position to another, avoiding the opponents.

In order to implement the strategies, and tactics, CAMASS provides a wrap allowing the visualization of the logic of every decision level. Currently there is the implementation of some simple tactics.

V. RESULTS

The offensive plays are implemented, based on the abilities already existing in the published code from WrightEagle. These abilities are: (1) **Kicker**, who calculates and takes the necessary measures in order to send the ball to the desired position; (2) **Dribler**, who is in charge of taking the ball to a given position, whenever the player has control of the ball; (3) **Dasher**, who allows a player to go a given position without losing the ball sight. The formal definition of plays dribble, shoot and goto are:

\[
\text{dribble} \equiv \text{dasher(ballPosition)} \rightarrow \text{dribler(position)}
\]
\[
\text{shoot} \equiv \text{dasher(ballPosition)} \rightarrow \text{kicker(position)}
\]

where ballPosition is the current ball position and position is the aimed position.

\[
\text{goto} \equiv \text{dasher(evasionPoint)} \rightarrow \text{dasher(position)}
\]

where evasionPoint is an intermediate position to evade another, mate or opponent player.

Using these basic plays, it is possible to define Simple, as well as Complex, Tactics. Some examples are:

A. **Pass Simple Tactic**

Player 1 has the ball and kicks it to a given position where player 2 controls it (Fig. 4). The plays coordination is given by explicit timing: At t1, player 1 advances and prepares the kick, player2 starts to run to final position; at t2, player1 kicks the ball and player2 is running to final position; at t3, player1 is at
another position (irrelevant for this tactic) and player2 has controlled the ball.

Figure 4. The Pass Simple Tactic schema between two players. Blue arrows represent the movement of a player while controlling the ball, dashed green arrows represent the movement of players without the ball and orange arrows represent the ball displacement.

Formally:
\[ s_{\text{pass}, \text{player1}} \equiv \text{dribble (position}_1\text{) }\rightarrow\text{shoot (position}_2\text{) }\rightarrow\text{goto (position}_3\text{)} \]
\[ s_{\text{pass}, \text{player2}} \equiv \text{goto (position}_2\text{) }\rightarrow\text{intercept ()} \]

where \( s_{\text{pass}, \text{player1}} \) are the plays for player1 and \( s_{\text{pass}, \text{player2}} \) are the plays for player2. The Pass Simple Tactic is expressed by the plays of both players:

Pass Simple tactic \((s_{\text{pass}, \text{player1}}, s_{\text{pass}, \text{player2}})\) \(\equiv s_{\text{pass}, \text{player1}}, s_{\text{pass}, \text{player2}}\)

B. One-two Simple Tactic

A little more sophisticated tactic is the One-Two, where player1 pass the ball to player2, who returns it to player1 in another position (Fig. 5). The main timing coordination occurs at \( t_1, t_3 \) and \( t_5 \)

Formally,
\[ s_{\text{one-two}, \text{player1}} \equiv \text{dribble (position}_1\text{) }\rightarrow\text{shoot (position}_2\text{) }\rightarrow\text{goto (position}_4\text{) }\rightarrow\text{intercept ()} \]
\[ s_{\text{one-two}, \text{player2}} \equiv \text{goto (position}_2\text{) }\rightarrow\text{intercept () }\rightarrow\text{dribble (position}_1\text{) }\rightarrow\text{shoot (position}_4\text{) }\rightarrow\text{goto (position}_5\text{)} \]

This tactic is expressed as:

One-Two Simple tactic \((s_{\text{one-two}, \text{player1}}, s_{\text{one-two, player2}})\) \(\equiv s_{\text{one-two}, \text{player1}}, s_{\text{one-two, player2}}\)

C. Withdrawal Simple Tactic

This is a defensive tactic where each team player must go to its own default position to mark the opponent team and try to recover the ball (Fig. 1). Initially this tactic is considered as the Panic tactic.

Formally:
\[ s_{\text{withdrawal, player1}} \equiv \text{goto (position}_{\text{default1}}\text{) }\]
\[ \ldots \]
\[ s_{\text{withdrawal, playerN}} \equiv \text{goto (position}_{\text{defaultN}}\text{) }\]

Withdrawal Simple tactic \((s_{\text{withdrawal, team}}) \equiv s_{\text{withdrawal, player1}}, s_{\text{withdrawal, player2}}, \ldots, s_{\text{withdrawal, playerN}}\)

D. Deep Right Attack Complex Tactic

This is an example of a complex tactic based on simple tactics above (Fig. 6). Four players intervene to finally shoot to opponent goal. Player3 should decide to play one-two simple tactic with player4 or with player5, depending on opponents’ location.

Formally (brackets [ ] denote alternative plays or tactics to execute):
\[ c_{\text{DRA, player1}} \equiv s_{\text{pass}, \text{player1}}, \text{ player2} \]
\[ c_{\text{DRA, player2}} \equiv s_{\text{pass}, \text{player1}}, \text{ player2} \rightarrow s_{\text{pass}, \text{player2}}, \text{ player3} \]
\[ c_{\text{DRA, player3}} \equiv s_{\text{pass}, \text{player2}}, \text{ player3} \rightarrow [s_{\text{one-two, player3}}, \text{ player4}, s_{\text{one-two, player3}}, \text{ player5} ] \rightarrow \text{shoot (goal)} \]
\[ c_{\text{DRA, player4}} \equiv [s_{\text{one-two, player3}}, \text{ player4}] \]
\[ c_{\text{DRA, player5}} \equiv [s_{\text{one-two, player3}}, \text{ player5}] \]

This tactic is expressed as:
E. Simulation of a complex tactic in the Robocup platform

This simulation over the monitor of simulated 2D server systems shows a complex offensive tactic based on two players who try to run side-by-side (Fig. 7). Player1 dribbles the ball as far as possible while player2 dashes. If player1 considers being close to the opponent goal, he shots the ball. Otherwise, in case of encounter an opponent, player1 makes a pass to player2. In both cases, this tactic finishes. If player2 receives the ball, he can also apply this tactic with player1.

To deal with the testing issues of the architecture, the language DLCAMASS is proposed to facilitate the definition of test case scenarios from disk files: the CAMASS decision levels can be tested in specific scenarios without changing their implementation. This mechanism improves the testing process effectiveness and consequently the whole process of the architecture implementation. In competition, this mechanism allows to adapt/change easily the team strategy, increasing then the probability of winning.

In order to complete CAMASS implementation, the following improvements are necessary: the improvement of tactics and strategies sets, the definition of a coach, who will select the appropriate team strategy, and the completion of communication protocol. Additionally, it is desirable a graphical system to define strategies and tactics.

Finally, a deeper study of plays, medium and long horizon planning used in human soccer, should increase the effectiveness of upper decision levels and therefore the probability of winning in robotic soccer championships.

VI. CONCLUSIONS AND FUTURE WORK

Robotic soccer is a challenging problem because of its dynamic nature and the endless space of possible plays. The most important aspects of a successful team structure include cooperation, multiple decision levels and testing. CAMASS proposes an architecture to support, in a structured way, the specification, implementation, loading and testing of robotic soccer plans. Based on solutions already existing in human soccer, CAMASS defines five decision levels, each one with its own horizon time of plans, where higher levels promote cooperative team actions and long term horizon time of plans: Strategy considers the more general game issues as the score and the opponent team’s attitude. Complex tactics consider the position of players and the ball possession, promoting the reuse of simple sequences of plays in new, more complex, sequences. Simple tactics consider the position of the ball and of some players, in order to execute a short sequence of plays. Play is in charge of performing a reactive planning depending of some parameters. Skill allows the use of the system in different robotic platforms. The current CAMASS’s implementation is based on the architecture’s lower decision levels (Skill, Play, and Simple tactic) considering instant, short and medium term horizon time of plans.

REFERENCES