Relational Behaviors For Soccer Robots

Quantitative Analysis

Dissertação para obtenção do Grau de Mestre em Engenharia Electrotécnica e de Computadores
Extended Abstract

João Gonçalo Delgado Torres Torres
Instituto Superior Técnico
Supervisor: Prof. Pedro Lima
Co-Supervisor: Engº Rodrigo Ventura

***Abstract*—The goals of this Thesis are the development, demonstration and analysis of relational behaviors, developed in the scope of SocRob project, for static situations of game, for instance the goal kick. The behaviors were designed using Petri Nets and its implementation followed an algorithm modified from previous work. As an essential requisite to a relational behavior, mechanisms of commitment establishment and management as well as of synchronization were implemented. Using the same code, the behaviors were tested in the simulator and in the real robots. Finally, a quantitative analysis of the developed Petri nets was carried through.**

***Keywords*—Relational Behaviors, Commitment, Synchronization, Petri nets, Robotic Soccer, Quantitative Analysis**

I. Introduction

This document summarizes the work developed for this project and is organized in the following way: this section presents the motivation, work environment and related work. The work background is introduced in Section II. In Section III the work and some aspects of implementation are explained and in Section IV the set of implemented behaviors is summarized. Finally, in Section V the results are presented and in Section VI the conclusions.

A. Motivation

Any society tends to organize itself to complete tasks that would be impossible to be completed by just one individual. Robotics creates populations of robots that when functioning as one distributed robot reach objectives that would be difficult to achieve if the task was assigned to only one robot. We can observe this concept in space exploration, automobile industry, search-and-rescue robots and robotic soccer.

Robotic soccer, like human soccer, consists on the confrontation of two teams whose players cooperate to defeat the opponent team. Again, like in human soccer, the team with better results is the team in which the players cooperate the most. This project is centered on the development of relational behaviors for one team of soccer robots. In order to reach the objectives and in any task that involves several robots, they need to work well together. This request raises the main problems related with the implementation of relational behaviors. Human beings are endowed with sensorial functions that allow them to communicate in an implicit (where the player takes decisions from what he observes) and explicit way (where the player takes decisions based on messages). Since the implicit way is still far from being implementable in robotics, explicit communication was used here.

Vecht[5] defines a relational behavior as a set of individual behaviors that are to be executed in a coordinated manner by a set of agents from a cooperative team. The participants pursue a joint goal and communicate with each other to achieve the required coordination. This implies an agreement between the participants, referred to as a commitment.

B. Work Environment

This work was developed in the scope of SocRob[1][7] project at the Institute of Systems and Robotic. The acronym of the project stands both for "Society of Robots" and "Soccer Robots", the case study where its population of five robots is being tested. The project participates regularly in competitions, like RoboCup, playing in Middle-Size League (MSL).

Currently, the field has a dimension of 18x12 meters, green color, white lines and the goals are distinguished by the yellow and blue colors. In the corners there are yellow and blue poles. The ball must be orange.

A simulator that recreates the MSL environment was developed in the scope of SocRob project, using Webots[9]. This application is very important in the development of behaviors, since it allows to reduce the development time. This simulator runs the exact same code that runs in the robots.

C. Related Work

The relational behaviors implementation is supported by Joint Commitment Theory[2] presented in section II-C. This theory demands that a commitment should be establish between the agents involved in the behavior.

[1] http://socrob.isr.ist.utl.pt/

To break such a commitment the acknowledgment of the robot that is going to finish is needed.

The development of applications based on the Joint Commitment Theory, including communications, has been extensively reported by Tambe[11]. One example is Yokota[10] et al. whom use explicit communications to achieve cooperation and synchronization between real robots. Research in the Simulation League of RoboCup have achieved cooperative behaviors among virtual agents, either executing a pass through implicit communications (observation of the team mates behavior)[12], or through the learning by a neural network [13]. However, no commitment between players is previously established in the works mentioned above.

Vecht[5] implemented relational behaviors integrating commitment. All the behaviors were designed with three phases: setup, loop and end. Each phase is divided in some states and the synchronization between robots is reached guaranteeing that all robots find themselves in the same state. However, despite the good results this implementation had, it forces the developer to define all the states in order to achieve synchronization, which, in turn, increases the development time. The formalism used to model behaviors was state machines which are less powerful than Petri nets. Using state machines, for instance, it is impossible to use parallelism.

II. Background

A. Petri Nets

Petri nets[3] are a useful tool in the description of discrete event systems.

A Petri net is a graphical and mathematical modeling tool, consisting of places, transitions, and arcs that connect them. Input arcs connect places with transitions, while output arcs start at a transition and end at a place. Figure 1 is an example of a Petri net.

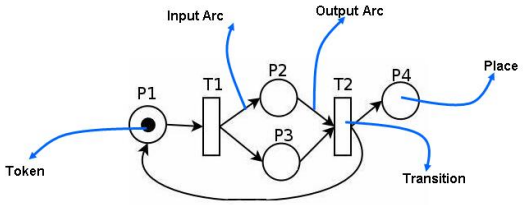

Figure 1. Petri net Example.

Places can contain tokens, represented by points or numbers. The current state of the modeled system, called the marking, is given by the number of tokens in each place at a precise moment. To fire a transition the input place must have as much tokens as the weight of the arc that connects the place and the transition. A transition can be associated with an event. In this case, it only fires if it is enabled and the event has occurred. The number of tokens removed / added depend on the weight of each arc.

(a) Before firing (b) After firing

Figure 2. Example of Petri net transition firing

B. Stochastic Petri Nets

N. Viswanadham and Y. Narahari.[1] introduced the stochastic Petri nets that were used for analysis in this work. A Generalized Stochastic Petri net(GSPN) adds time and probability to a Petri net allowing these nets to perform quantitative analysis. GSPN uses two transition types: immediate and exponential. Firing times for an exponential transition are exponentially distributed. If two or more exponential transitions are enabled in the marking M_i and T_i denotes that particular set of transitions, the probability for transition t_j fire is:

$$\frac{F(M_i, t_j)}{\sum_{t_k \epsilon T_i} F(M_i, t_k)} \quad (1)$$

where $F(M_i, t_j)$ is the rate of transition t_j in the marking M_i. Naturally, a transition with a higher rate has a bigger probability of being chosen to fire. An immediate transition is represented by a line and an exponential transition by a rectangle.

C. Joint-Commitment Theory

For Cohen and Levesque[2] coordination in multi agent systems and teamwork are different concepts. They gives an example: "while common traffic is clearly a coordinated system, it is not teamwork. Driving in a convoy however, is seen as teamwork, because the involved agents share a common goal and need to cooperate in order to achieve their goal". The Joint-Commitment Theory is supported by the idea that agents with a common goal should join in a commitment. They inform each other about their progress to achieve the goal. Applying this concept to soccer robot leads to the following conclusion: once a robot is committed to a relational behavior, he will pursue the execution success until some conditions become false or until the goal has been achieved. The behavior success or failure has to be reported to all participants in the commitment before the agent quits the commitment. This way, all the robots will be informed all time and will not continue to pursue impossible objectives.

D. Behaviors

A behavior is a set of actions that carried through in parallel or in series has the objective of executing

one determined task. The sequence of actions depends on conditions that are related to the robot and to its surroundings. One behavior example is the task to take the ball until the goal. It contains actions such as catching the ball, dribbling and kicking. The execution moment of each action is defined by the surrounding conditions: the robot will not be able to kick the ball if it is not in its possession.

Behaviors can be classified using three classes:

- **Organizational Behaviors:** this type of behaviors is responsible for the team organization in the field and attributes a role to each player(defender, attacker, supporter).
- **Relational Behaviors:** behaviors where two or more robots are involved to complete a task (to pass the ball).
- **Individual Behaviors:** behaviors that involve only one agent (to cover the goal, to support the attack, to attack).

The choice of which type of behavior must be carried through at each moment depends on the team's software architecture.

E. MeRMaID

Previously to the work presentation it is important to describe in summary the software architecture developed in the project and that supports the behaviors and features implemented. This architecture is described[6] as: "MeRMaID (Multiple- Robot Middleware for Intelligent Decision-making) is a robot programming framework whose goal is to provide a simplified and systematic high-level behavior programming environment for multi-robot teams, which simultaneously constrains some of the developer options, so as to guide him/her towards building better and maintainable code."

The MeRMaID framework defines the entities:

- **Roles:** are subsets of behaviors, defined over the set of available behaviors (e.g., Attacker, Defender, Supporter);
- **Behaviors:** are defined as "macros" of primitive actions grouped together using some appropriate representation, in this case Petri nets.
- **Primitive Action:** is the atomic element of a behavior, which can not be further decomposed. It is designed as a STA (Sense-Think-Act) loop.
- **Macro:** with this feature the user can reuse some sets of behaviors. Two places tagged with GOAL_REACHED and GOAL_NOT_REACHED should be defined to identify the execution success or insuccess;
- **Predicates:** are Boolean relations over the domain of world objects, e.g., seeBall;
- **Event:** is an instantaneous occurrence which denotes a state change.

Behaviors are implemented using three levels:

- **Team organizer:** attributes one role to each robot;
- **Behavior Coordinator:** chooses the behavior the robot should execute;
- **Behavior Executor:** chooses the action that the robot should execute;

MeRMaID also provides a framework that executes Petri nets. In order to run Petri nets using this framework, the nets should follow some rules:

- **Places**
  - **Tag <action>**: at TeamOrganizer level indicates a role, at BehaviorCoordinator level indicates a behavior and at BehaviorExecutor level indicates a primitive action;
  - **Tag <predicate>**: indicates a predicate;
  - **Tag <macro>**: indicates a macro;
  - **Place with no tag**: indicates a memory place;
- **Transitions**:
  - **Tag <event>**: indicates a transition associated with an event.
  - **Transition with no tag**: indicates a immediate transition with no association;

III. Representation of Relational Behaviors Based on Petri nets

A. Commitment Establishment

As mentioned before, the commitment grants each robot the knowledge that it is part of a relational behavior that involves more teammates. The commitment establishment crosses some stages until robots are committed. Below, the solution developed and used in this work is presented, considering that robot R1 sends the request to establish a commitment with its teammates, here represented by robot R2:

- **R1 - Asked for establishment of the commitment:** robot R1 sends an invitation to the teammates to establish a commitment for any relational behavior;
- **R2 - Analysis and reply to the request from R1:** robot R2 receives the request and verifies if it is in conditions to enter in the commitment. In the affirmative case, it sends a message in return. Then it waits for the answer to its candidacy;
- **R1 - Analysis and answers to the candidacies:** robot R1 receives the answers to his request during a certain period. After this period of time, it analyzes them and answers with an accept or reject. It then waits a commitment message from R2. If there were no answers to its request it is considered not engaged;
- **R2 - Reception of the reply to the candidacy:** robot R2 waits its answer. In the case of a positive response, this means that it will enter in the commitment with R1. In the negative case it means that it did not succeed. In the first case, R2 sends a message

indicating that it is committed. Finally, it waits for the same type of message from R1;

- **R1 - Reception of the commitment message:** robot R1 receives the commitment message from R2 and sends an identical one in return. At this point, robot R1 is committed.
- **R2 - Reception of the commitment message:** robot R2 receives the commitment message that it was waiting for and considers itself committed with R1.

To prevent a deadlock during the commitment establishment, the robots use timers. This way they will not wait a message forever. The Figure 5 is the Petri net developed to establish the commitment between the robots and uses the method explain above.

B. Commitment Management

A relational behavior execution ends with its success or failure. In both cases, the robot(s) intervening in the behavior must be in agreement about the state of the commitment, meaning that if one of the robots wants to break the commitment it must inform the other(s). This way none of robots will be blocked in a finished behavior. For this reason, it is necessary to manage the commitment.

Figure 4 depicts an Petri net example for a relational behavior while Figure 3 represents the respective commitment management[4]. The blue and green states of Figure correspond to the states of the same color in Figure 4 which means that each primitive action has its own commitment management Petri net.

Three places in a commitment management net are necessary:

- **Commitment:** token in this place indicates that the partner is engaged;
- **End Commitment:** this place receives one token when the behavior did not succeeded and therefore the conditions for the robot to continue in commitment do not exist;
- **NotCommitted:** this place indicates that the robot is not compromised, leading to the interruption of the primitive action. Token is placed here if the partner has failed in the behavior and has consequently broken the commitment.

These nets describe a possible implementation to manage the commitment. However, to adapt this method to MeRMaID, some changes had to be made. Instead of a constant management, as the one presented previously, the implemented behaviors only verify the commitment in some strategic points . These points are those from which the robot action would be significantly changed in case that it was not compromised. The point where the robot that makes the pass waits for the robot that will receive the ball is assumed as a point of this type. At

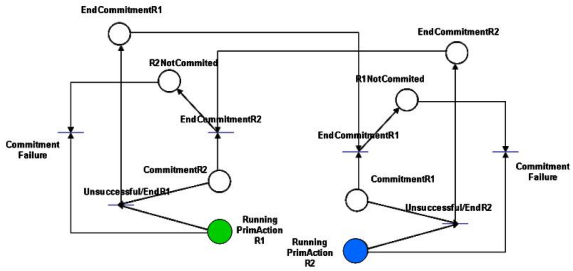

Figure 3. Petri net to manage commitment[4].

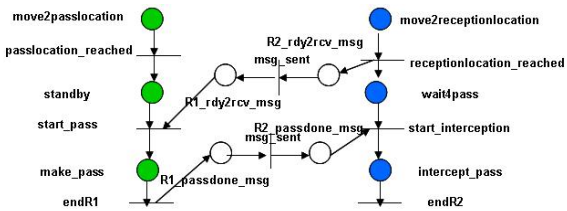

Figure 4. Relational behavior example[4].

these points as evaluation concerning the commitment is performed, using the predicate **PartnerCommited** whose value depends on the information transmitted by the commitment partner. If during an evaluation moment it verifies that the value of the predicate is false, the robot also breaks the commitment, given that the commitment was already broken by the other robot. The case cited above occurs when the commitment partner fails to execute the behavior. Still, in case of success or own failure the commitment must also be broken.

The option of implement a management that performs a not constant evaluation is supported by the fact that it increases the behavior performance. This type of approach requires higher attention in the development of the behaviors so that one higher performance in one robot does not have as consequence the behavior failure in its partner.

C. Synchronization

Another essential requirement to a relational behavior is synchronization. The sequence of actions that forms a task can be executed correctly by its intervening subjects but, if there is no synchronism between the robots, the insuccess is the more probable end. In a pass, if the robot passes the ball before the receiver is prepared, the team will lose it.

The synchronization was obtained through the transmission of messages meaning that explicit communication was used, as described in Section I-A. This fact must be taken in account because synchronism points are an easy target to communications problems. A behavior must not have many points of synchronism and none of them, in case of failure, can block the behavior execution.

The synchronization in the developed behaviors is necessary in two moments only: when the player that

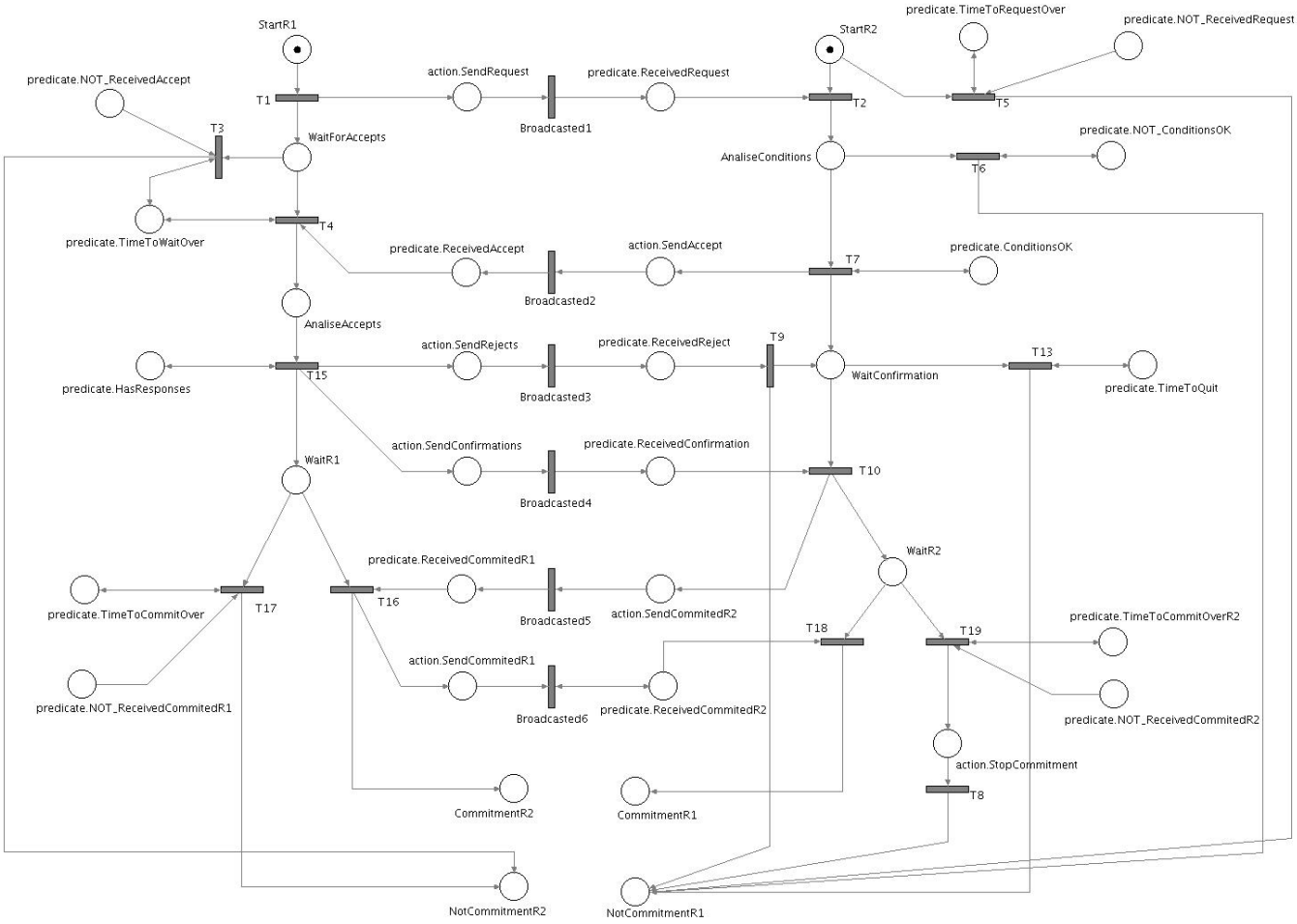

Figure 5. Petri net to establish the commitment.

goes to receive the ball wants to inform the taker that it is ready to receive and when the player that takes the foul wants to indicate that the ball was passed. To complete this implementation, events were used. Figures 6 and 7 present an example for a pass. The option of using events is justified for its implementation simplicity. A synchronism point represents a prompt situation that has no other objective than to inform the other robot about the fulfilment of the task at hand. No processing or information storage is performed. A memory place is in the Petri Nets used so, if the Petri net execution still does not have reached the synchronism point, it saves the token and the event will not be lost (ReceiverReady and PassDone in Figures 6 and 7).

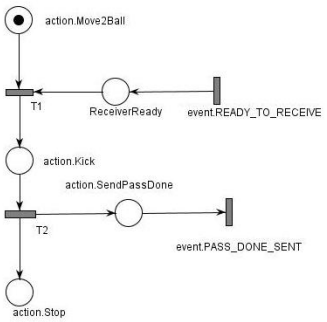

Figure 6. Example of synchronization. Robot that passes the ball.

D. Implementation Algorithm

An algorithm was used to achieve a faster and methodical way of implementing behaviors . The said methodology is composed by the following steps:

1) Identification of the behavior objective;
2) Petri net drawing;
3) Identification of the synchronism points, survey of the events and necessary timeout mechanisms;
4) Identification of the points where is necessary to manage the commitment
5) Identification of the primitive actions, predicates and events needed in the new behavior and coding

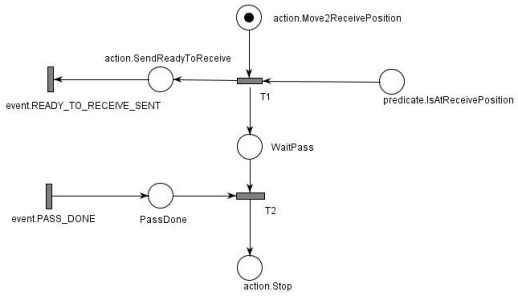

Figure 7. Example of synchronization. Robot that receives the ball.

of the new ones;

6) Behavior tests, running it first using Webots and then real robots.

With this algorithm the drawing of new behaviors was automatized promoting an easy work organization in the creation of new primitive actions, predicates and events, for instance.

IV. Implemented Behaviors in Soccer Robots

This section presents the behaviors that were implemented during this work. Individual, relational and organizational behaviors were developed. The behaviors are organized in three groups, each one corresponding to one MeRMaID level: TeamOrganizer, BehaviorCoordinator and BehaviorExecutor.

A summary about the tactic developed and executed at the TeamOrganizer level is presented here. The follow roles were implemented:

- **RoleAttacker:** This role implements the team attacker and is executed by the robot closest to the ball;
- **RoleDefender:** This role implements the team defender and is executed by the robot closest to its own goal that is not the one closest to the ball;
- **RoleSupporter:** The organizational behavior that is executed by the robot that supports the attacker is implemented by this role. The TeamOrganizer chooses the robots that were not elected to attacker or defender;
- **RoleFoulTaker:** In case of ball throw-in, goal kick, free kick, corner kick, penalty or kick-off for own team the robot closest to the ball is chosen to take it;
- **RoleFoulReceiver:** The ball is passed to the robot that executes this role during a ball throw-in, goal kick, free kick, corner kick, penalty or kick-off. The robot chosen is the second closest to the ball.

Each role executed by the BehaviorCoordinator can be described in a concise way:

- **RoleAttacker:** This role implements the attacker. In case of ball throw-in, goal kick, free kick, corner kick, penalty or kick-off, the attacker maintains a certain distance to the ball; either way, its objective is to score a goal;
- **RoleDefender:** The main objective of the defender is to protect the goal. If the ball is near its goal, it covers the angle between them; either way, the robot stays near its goal, waiting opponent's next attack;
- **RoleSupporter**: The supporter follows the attacker. This way, if it looses the ball, the supporter, nearer to the ball, can gain its possession again;
- **RoleFoulTaker:** This role is executed by the robot that repositions the ball. It chooses between a set of foul taker behaviors, depending on the game situation and, for each situation, on a random variable;
- **RoleFoulReceiver:** This role is executed by the robot that receives the ball in a reposition. It chooses between a set of foul taker behaviors, depending on the behavior chosen by the taker.

The behaviors executed by the BehaviorExecutor are explained below. The individual behaviors implemented were:

- **BehaviorBaseAttack:** This is the attacker's main behavior. Its objective is to score goal. It uses two different opponent's goal approaches and chooses between them using a random variable;
- **BehaviorBaseDefend:** This is the defender's main behavior. Its objective is to cover the angle between the goal and the ball;
- **BehaviorBaseSupport:** The supporter executes this behavior to search for the ball and then to maintain a certain distance;
- **BehaviorFoulAttacker, BehaviorFoulSupporter, BehaviorFoulDefender:** These behaviors are executed by the attacker, the supporter and the defender, respectively, during a ball throw-in, goal kick, free kick, corner kick, penalty or kick-off. Their objective is to maintain a certain distance to the ball. The attacker stays closest to it, then the supporter, then the defender.

Finally, the relational behaviors developed are presented:

- **BehaviorFoulTakerDirect:** This behavior is executed by the taker. It envolves commitment between the taker and the receiver. Varying the robot final posture, four more relational behaviors to reposition the ball were implemented. As an example of this different postures we have to touch the ball or to pass through the line. The Petri net designed for this behavior is showed on Figure 8.
- **BehaviorFoulReceiveDirect:** It executes the corresponding behavior to BehaviorFoulTakerDirect execute by the receiver. Four more relational behaviors to receive the ball were implemented, varying the robot posture, just like mentioned before. Figure 9 shows the Petri net designed for this behavior.

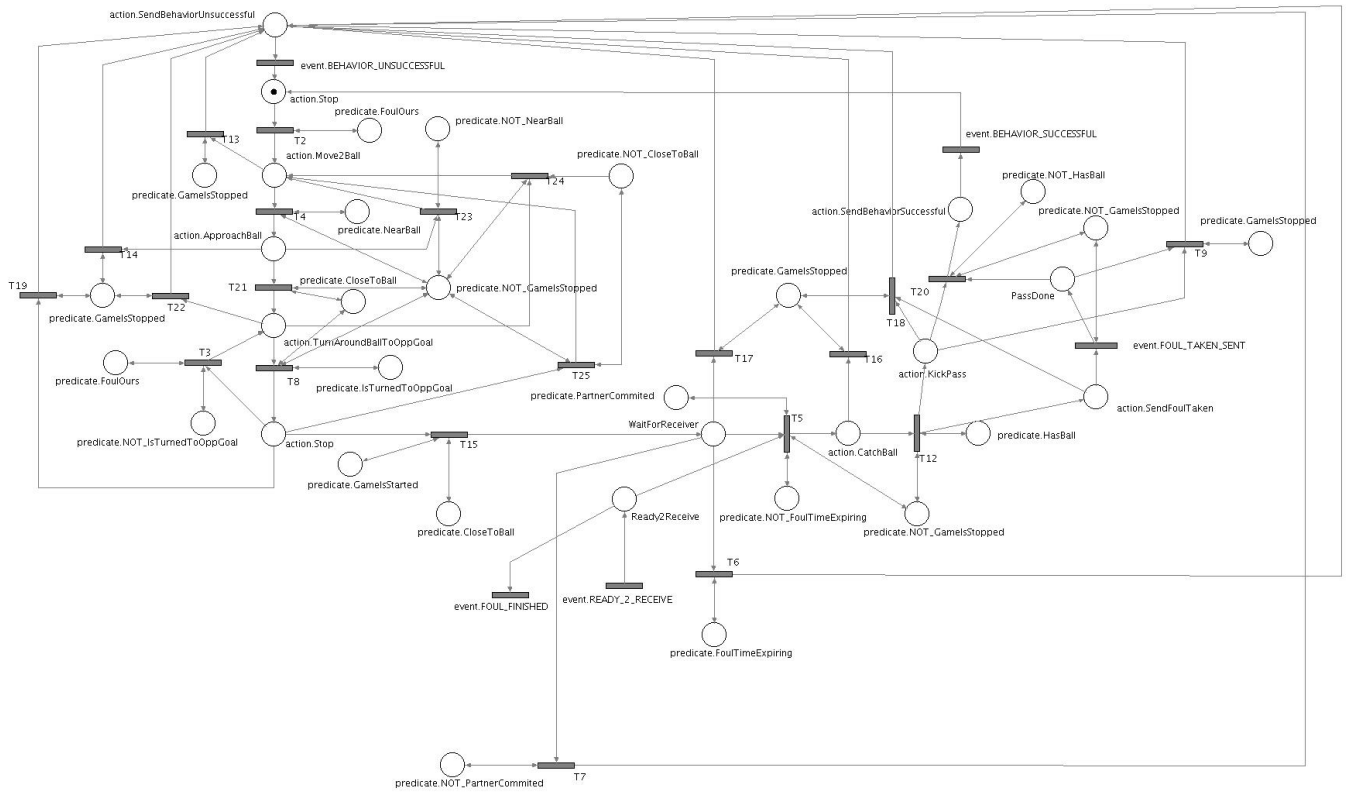

Figure 8. Petri net example of one behavior executed by the taker.

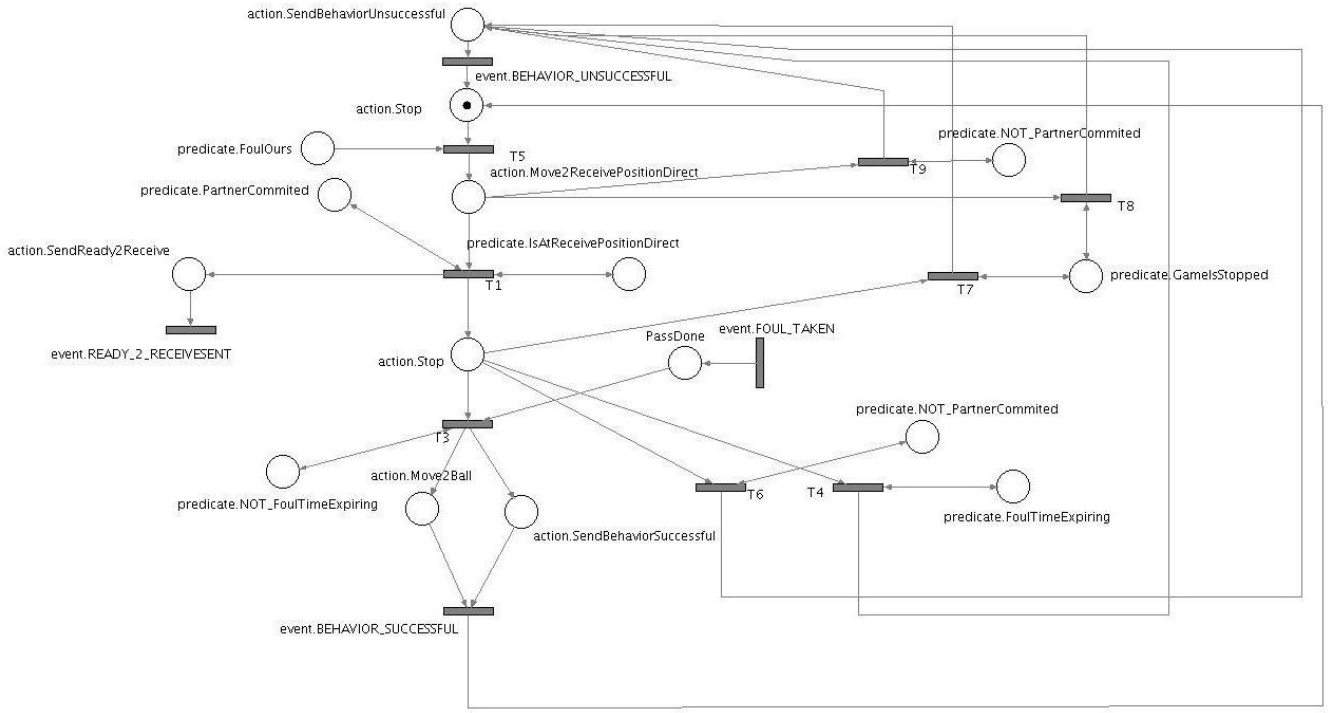

Figure 9. Petri net example of one behavior executed by the receiver.

V. Results and Analysis

A. Results

The results were satisfactory: the robots executed all the behaviors, both in the simulator and in the real world, both with success. Many of the developed behaviors were used by the project in the competitions that ISocRob team integrated during the development period of this work: Robotica 2007 and RoboCup 2007. Some videos of the robots executing the behaviors can be viewed in the site:

http://socrob.isr.ist.utl.pt/videos/behaviors/behaviors.html

As previously mentioned, the code that runs in the simulator and in the real robots is the same.

B. Analysis

The quantitative analysis done had as main objective find answers to the questions below:

- How does the primitive action failure affects the behavior failure?
- How should the primitive actions perform to achieve a determined behavior performance?
- Can a primitive action have greater impact on the behavior success than the others?
- What is the impact of an unsuccessful primitive action on the failure of the behavior?
- What behavior does the number of primitive actions have in the behavior success?

The analysis used the tool TimeNet[8]. For each primitive action in the analyzed behavior it considered two hypotheses of outcome, success and failure, which were simulated using two exponential transitions. Varying its rate different success and insuccess probabilities were used in this analysis. The consecutive success of each action indicates the behavior success and the unsuccess on any action has as consequence the behavior unsuccess. The results substantiated that to achieve successful behaviors the primitives should be almost perfect, specially when the behavior involves many actions. They also confirmed that the communication quality limits the behavior quality, given that, if some robot fails when sending a message the behavior will not be well succeeded.

Figures 10 and 11 and illustrate two examples for the said analysis. Figure 10 shows the results from the analysis made to answer the question: "How does the primitive action failure affects the behavior failure?". The behavior used to obtain it was the BehaviorBaseAttack, described in Section IV. The graphic evidences that to achieve a reasonable behavior performance (in the order of 80%) very good primitive actions are needed (with a rate performance near 95%), and to achieve an high behavior performance (in the order of 90%) almost perfect primitive actions (achieving 98% of success) are needed.

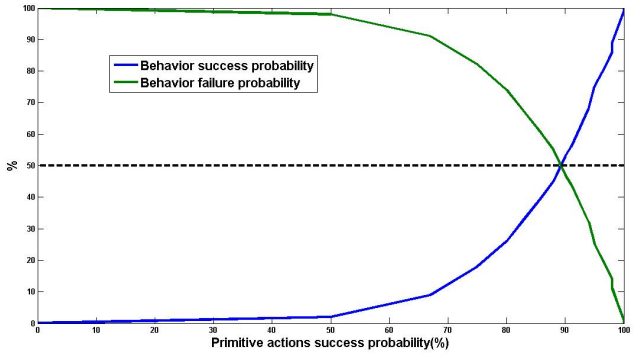

Figure 10. Impact of the primitives actions performance in the behavior performance.

The graphic in Figure 11 shows the impact of the primitive actions number on the behavior performance. The behaviors used in this simulation are composed by primitive actions series. As expected, the behavior success will increasedly depend on the primitive actions performance as more of those are present. The behavior developer should first invest on improving primitive actions before increase the behavior complexity.

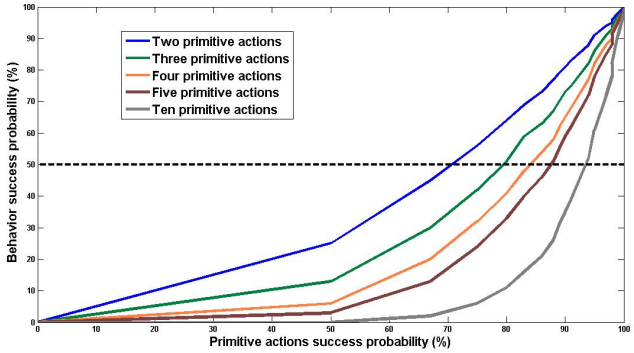

Figure 11. Impact of the number of primitives actions in the behavior success.

VI. Conclusions and Future Work

This extended abstract summarizes the work developed in the scope of this Thesis. The goals of this work were the development, demonstration and analysis of relational behaviors, developed in the scope of SocRob project, for static situations of game. The goal kick is one of this situations. The behaviors were designed using Petri nets and its implementation followed an algorithm. As an essential requisite to a relational behavior, mechanisms of commitment establishment and management, as well

as of synchronization, were implemented. Finally, a quantitative analysis of the developed Petri nets was carried through.

This work substantiates that robotic soccer should invest new efforts to develop better relational behaviors. This fact is not only justified by competitive reasons but by scientific reasons, as well, given that this work can be used in different areas. As future work in this interesting area, the SocRob project may implement dynamic passes. In order to achieve more reliable relational behaviors, the communications should be improved.

Also in this area, we could study decision systems that choose who should be involved in a commitment. It would be interesting to make a study using the diverse political systems to make this decision. This analysis would have as extremes an absolutist approach, where one robot would take all these decisions, and a democratic system, where each robot takes equal part in the decision.

REFERENCES

[1] N. Viswanadham and Y. Narahari., *Performance Modeling of Automated Manufacturing Systems*, Prentice Hall, New Jersey, 1992

[2] P. R. Cohen and H. J. Levesque. Teamwork. Nous, 35:487-512, 1991

[3] T. Murata, *"Petri Nets: Properties, Analysis and Applications"*, an invited survey paper, Proceedings of the IEEE, Vol.77, No.4 pp.541-580, April, 1989

[4] Pedro U. Lima, *Models of robotic tasks based on discrete event and hybrid systems*, Slides of Presentation, 2006

[5] B. Vecht and P. Lima, Formulation and Implementation of Relational Behaviours for Multi-Robot Cooperative Systems, 2004

[6] Marco Barbosa, Nelson Ramos, Pedro Lima, *"MeRMaID - Multiple-Robot Middleware for Intelligent Decision-Making"*, Proc, of IAV2007 - 6th IFAC Symposium on Intelligent Autonomous Vehicles, Toulouse, France, 2007

[7] Projecto SocRob, *SocRob Project - Description*, http://socrob.isr.ist.utl.pt/

[8] Zimmermann, A.; Freiheit, J.; German, R. Hommel, G.: *Petri Net Modelling and Performability Evaluation with TimeNET 3.0.*, 11th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS'2000), LNCS 1786, pp. 188-202, http://pdv.cs.tu-berlin.de/ timenet/, Springer-Verlag, Schaumburg, Illinois, USA, 2000

[9] WEBOTS, http://www.cyberbotics.com

[10] K. Yokota, K. Ozaki, N. Watanabe, Akihiro Matsumoto, D. Koyama, T. Ishikawa, Kuniaki Kawabata, Hayato Kaetsu, Hajime Asama, "Cooperative Team Play Based on Communication", RoboCup 1998 Book, Springer-Verlag, Berlin 1999

[11] M. Tambe, "Towards Flexible Teamwork", Journal of Artificial Intelligence Research, 7:83- 124, 1997

[12] E. Pagello A. D'Angelo, F. Montsello, F. Garelli, C. Ferrari, "Cooperative Behaviors in Multi-Robot Systems Through Implicit Communication", Robotics and Autonomous Systems, 29:65-77, 1999

[13] H.Matsubara, I. Noda and K. Hiraki, "Learning of Cooperative Actions in Multi-Agent Systems: a Case Study of Pass in Soccer", Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the AAAI-96 Spring Symposium, SS-96-01, pp63-67, 1996