ISocRob-4LL 2006

Team Description Paper

Pedro Lima, Marco Barbosa, João Esteves, Nuno Lopes, Vasco d'Orey, Hugo Pereira
Institute for Systems and Robotics, Instituto Superior Técnico
Av. Rovisco Pais, 1049-001 Lisboa, PORTUGAL

1 Introduction and Overview

The ISocRob team and its regular participation in RoboCup since 1998 are the competition side of the SocRob project[1], a research endeavor of the Intelligent Systems Laboratory of the Institute for Systems and Robotics at *Instituto Superior Técnico* (ISR/IST), Technical University of Lisbon, which started in 1997. The project goal is to develop a novel approach to the design of a population of cooperative robots based on concepts borrowed from Systems Theory [3] and Distributed Artificial Intelligence [6]. ISocRob has participated in RoboCup Middle-Size League (MSL) in 1998, 1999, 2000 (European Cup), 2001, 2003 and 2004, as well as in the Soccer Simulation League in 2003 and 2004, and started a new Four-Legged League (4LL) team in 2005. We have pre-registered for MSL and 4LL this year.

For the SocRob project, RoboCup 4LL offers the advantages of a stable hardware platform, common to other teams, so that a new team can start from existing hardware and software. Furthermore, our regular participations in the MSL have taught us many lessons concerning the desirable features of functional and software architectures for cooperative robot teams, and we want to test our concepts by using a common functional architecture in the two leagues, as well as developing a software architecture shared to the maximum possible extent by the MSL and 4LL robots.

Our approach to the development of a 4LL team as a case study for testing some of our research interests was as follows:

1. Select, among the available code from other teams, the one closest to our architecture concepts, modular enough to allow building up new code without having to modify (at least substantially) the initial basis.
2. Adapt the code to fit our functional architecture concepts.
3. Use, from the existing code, most of the work concerning *individual* behaviors, *self*-localization, *individual* ball (and other objects) localization.
4. Develop new concepts and code concerning *cooperation* and *teamwork*, mainly *cooperative navigation* (mutual localization of teammates, formation control), *cooperative object localization* (e.g., using our past work on *sensor fusion* [8], *cooperative plan execution*.

In this Team Description Paper, we will describe our reference functional and software architectures, its implementation from available code, and the steps already taken towards building a cooperative robot team.

[1] The acronym of the project stands both for **Society of Robots** and Soccer Robots, the case study where we are testing our population of robots.

2 Functional and Software Architectures

The basic functional architecture of the SocRob team is organized in three levels of team member responsibility, according to the relations among robots they imply, similar to those proposed in [5]:

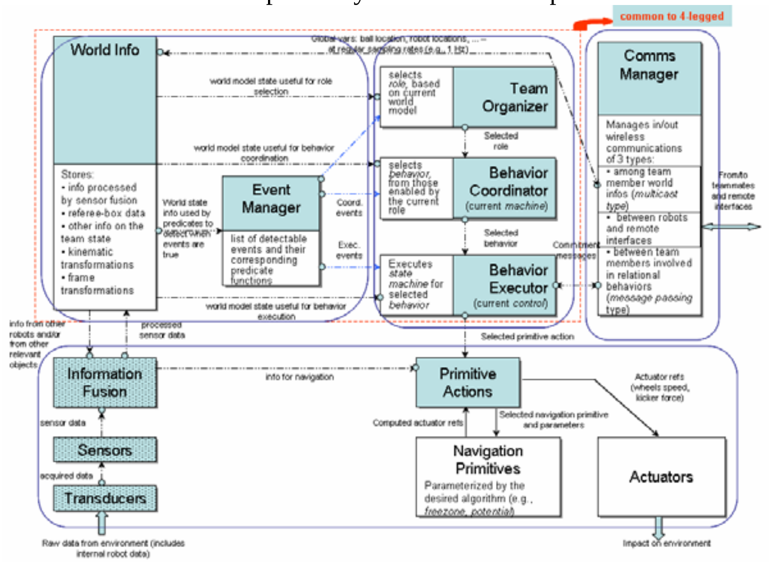
- **Organizational**: Those which are related to the team's organization, that is, the assignment of roles to players.
- **Relational**: Those involving two or more teammates, such as performing a pass or coordinately defending the goal.
- **Individual**: Those which are executed by one single robot.

Since behaviors are externally displayed and emerge from the application of certain operators, the functional architecture can also be viewed from an operator standpoint, with three levels of decision:

- **Team Organization Level**, where, based on the current world information, a *strategy* (i.e., *what to do*) is established, including a *goal* for the team. This level considers issues such as modeling the opponent behavior to plan a new strategy. Strategies may simply consist of enabling a given subset of the operators at each robot, in result of role assignments to each team member. In robotic soccer, basic roles can be `Goalkeeper`, `Defender`, `Attacker` and `Full Player` (i.e., simultaneously `Defender` and `Attacker`). Only the captain robot will have the organization level enabled. Should the captain "die", the next robot in a pre-specified list will have its organization level enabled and become the captain.
- **Behavior Coordination Level**, where switching among operators, both individual and relational, occurs so as to coordinate behavior execution, at each robot and among the team robots, towards achieving the team goal, effectively establishing the team *tactics* (i.e., *how to do it*). We have used finite state automata and a rule-based system before in the MSL team to implement this level, but other alternative representations are possible, such as Petri nets [2].
- **Behavior Execution Level,** where primitive actions run and where they interface the sensors and the actuators. Primitive actions are linked to each other to implement an operator. Currently, every operator (representing a given behavior) is implemented as a finite state automaton whose states are the primitive actions and transitions are associated to events that are detected by the system. Behaviors can be individual, if their corresponding operators run in one robot only, or relational, if two or more robots are running operators that are coordinated through commitments and synchronization messages to achieve a common goal (e.g., to pass a ball, to avoid moving simultaneously towards a ball, to cover a field region while the teammate advances in the field through role exchanges). As a result of the latter, a relational behavior is displayed.

The software architecture is the practical implementation of the functional architecture, which could be done in any programming language and using different software technologies.

The new software architecture of the SocRob project, depicted in Fig. 1, keeps the fundamental principles of the functional architecture the group has been following since 1999 [7]. Nevertheless, it was completely re-designed to be re-written on C++, as well as to improve the matching between software modules and the main functional concepts. The *decision-making* modules of the architecture are shared between the 4LL and MSL code. Communications-, sensor- and actuator-related modules can not be shared due to their inherent specificity and hardware dependence.



common to 4-legged
World Info
Global vars: ball location, robot locations, ... – at regular sampling rates (e.g., 1 Hz)
world model state useful for role selection
world model state useful for behavior coordination
Stores:
• info processed by sensor fusion
• referee-box data
• other info on the team state
• kinematic transformations
• frame transformations
World state info used by predicates to detect when events are true
Event Manager
list of detectable events and their corresponding predicate functions
Coord. events
Exec. events
world model state useful for behavior execution
selects role, based on current world model
Team Organizer
Selected role
selects behavior, from those enabled by the current role
Behavior Coordinator
(current machine)
Selected behavior
Executes state machine for selected behavior
Behavior Executor
(current control)
Commitment messages
Comms Manager
Manages in/out wireless communications of 3 types:
• among team member world infos (multicast type)
• between robots and remote interfaces
• between team members involved in relational behaviors (message passing type)
From/to teammates and remote interfaces
info from other robots and/or from other relevant objects
processed sensor data
Selected primitive action
Information Fusion
info for navigation
Primitive Actions
Actuator refs (wheels speed, kicker force)
sensor data
Computed actuator refs
Selected navigation primitive and parameters
Sensors
acquired data
Transducers
Navigation Primitives
Parameterized by the desired algorithm (e.g., freezone, potential)
Actuators
Raw data from environment (includes internal robot data)
Impact on environment

Fig. 1 – SocRob project new software architecture.

A brief summary of the main architecture modules follows.

The *World Info* is an object which stores the relevant information about the world, such as robot postures, ball position or current score. This information can be obtained either by sensor information or messages received from teammates or the referee box.

Table 1. Correspondence between software architectures

Our Team	**German Team**
World Info Incoming Messages	XABSL Input Symbols
Outgoing Messages	XABSL Output Symbols
Primitive Actions	XABSL Options

The *Behavior Executor* decides which primitive action to execute at each step, given a selected behavior. World Info data is used to take the decisions. Events are determined from World Info data as well, and are used to trigger internal state changes in the finite state automaton implementing a behavior. The *Behavior Coordinator* selects which behavior to run next, while the *Team Organizer* selects which *role* the player will perform. The selected *role* will only affect the set of behaviors a player can run, e.g., if a player has the role of defender, it will only be authorized to run defensive behaviors.

A decision was taken to use the German Team's architecture [12], as well as part of its code, and map it to the SocRob functional architecture. Table 1 lists the relevant correspondences. We have also used the Dutch Team 2005 patch [13], due to the improved cognition algorithms, as well as the updates for the new rules. Our `diff` is made against German Team 2004 code patched with Dutch Team 2005 `diff`.

3 Addressed Research and Future Challenges

As explained in the previous section, our main research challenges at the moment concern:

- **Cooperative Navigation**, where the knowledge about the localization of team members can be improved by mutual observation (e.g., as in [10] or [11]) – ongoing work. This topic will also include formation control, so as to distribute the robots across the field dynamically, e.g., by prescribing at each step the desired relative distances and orientations between teammates;
- **Cooperative Object Localization**, where the ball, other relevant objects (such as the poles around the field) and opponents are observed by more than one team member, and the resulting information is shared and fused using a Bayesian approach – ongoing work, mostly based on past developed work for the MSL team [8];
- **Cooperative Plan Execution**, mostly related to relational behaviors, where each involved robot executes part of the plan while regularly synchronizing it with the related teammates. This topic has been subject of major attention so far and will be more detailed in the sequel.

A Discrete Event Systems [3] based approach has been followed to the modeling of behaviors and their coordination. Individual behaviors are modeled by finite state automata (FSA), and some work has been done by the team on optimal task planning by composing primitive actions into FSA, given the uncertainty associated to the action effects and the goal of minimizing the time to a goal [9]. Due to their capability to model concurrency, Petri nets [2] are particularly suited for modeling relational behaviors, as they can capture the concurrent nature of exchanging messages between teammates while each of them is executing their own primitive actions. Petri net models have been developed based on concepts borrowed from Joint Commitment Theory [1][4].

The communication (either implicit, e.g., by mutual observation, or explicit, e.g., using communications) between the two players involved in the relational behavior is essential.

The messages sent by the two players can be of two different kinds:

- commitment related messages – to establish or break a commitment;
- behavior flow messages – to synchronize the execution of the behavior by both parties. In the *pass* example, the player who would receive the pass would send a message saying that it was waiting for the pass.

In our work, behaviors are modeled by PNs as follows:

- each place in the Petri net is labeled by an associated *primitive action* (e.g., `moving to a given posture`, `kicking the ball`, `intercepting the ball`) or resource (e.g., availability of an object, or of a robot, or of a communication signal);
- each transition in the Petri net is labeled by an *event*, defined in this context as occurring when a change of (logical conditions over) *predicate values* (from TRUE to FALSE or FALSE to TRUE) takes place, e.g., event `lost_ball` occurs when predicate `has(ball)` value changes from TRUE to FALSE.

A token in a place means that the primitive action associated to that place is currently active (i.e., it is running) or that the resource labeling that place is currently available. Transitions are enabled when all its input places have at least one token each, meaning that the pre-conditions for the next step are satisfied. A transition is fired if its enabled and the associated event occurs.

Acknowledgments

This work was supported by ISR/IST Pluriannual funding from the *Fundação para a Ciência e a Tecnologia* and POSI, in the frame of the EC FP6.

References

[1] P. Cohen, H. Levesque. “Teamwork”, *Nous*, Vol. 35, pg. 487-512. (1991)
[2] C. Girault, R. Valk, *Petri Nets for Systems Engineering*, Springer, (2003)
[3] S. Lafortune, C. Cassandras, Introduction to Discrete Event Systems, Kluwer Academic Publ., (1999)
[4] B. Van der Vecht, P. Lima, “Formulation and Implementation of Relational Behaviors for Multi-Robot Cooperative Systems”, *RoboCup-2004: Robot Soccer World Cup VIII*, Springer-Verlag, Berlin (2005)
[5] A. Drogoul, A. Collinot, "Applying an Agent-Oriented Methodology to the Design of Artificial Organizations: A Case Study in Robotic Soccer", *Autonomous Agents and Multi-Agent Systems*, Vol 1, pp. 113-129, Kluwer Academic Publ., (1998)
[6] J. Ferber, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Addison-Wesley, 1999
[7] P. Lima, L. Custódio, R. Ventura, P. Aparício, “A Functional Architecture for a Team of Fully Autonomous Cooperative Robots”, *RobCup-99: Robot Soccer World Cup III*, Springer-Verlag, Berlin (1999)
[8] P. Marcelino, P. Lima, “Bayesian Sensor Fusion for Cooperative Object Localization and World Modelling”. *Proceedings of the 8th Conference on Intelligent Autonomous Systems*, Amsterdam, The Netherlands (2004)
[9] B. Damas, P. Lima, “Stochastic Discrete Event Model of a Multi-Robot Team Playing an Adversarial Game”, *Proc. of 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles - IAV200*4, Lisboa, Portugal, (2004)
[10] S. I. Roumeliotis, G. A. Bekey. “Distributed Multirobot Localization”, *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, Ooctober 2002,
[11] D. Fox, W. Burgard, H. Kruppa, S. Thrun, “A Probabilistic Approach to Collaborative Multi-Robot Localization”, In Special issue of Autonomous Robots on Heterogeneous Multi-Robot Systems, 8(3), 2000,.
[12] T. Röfer et al, “German Team – RoboCup 2004”, Center for Computing Technology, Universität Bremen, pp. 23-116. (2004)
[13] J. Sturm, A. Visser, N. Wijngaards , “Dutch Aibo Team: Technical Report RoboCup 2005”