CoPS Stuttgart Team Description Paper 2003

T. Buchheim, G. Kindermann, R. Lafrenz, H. Rajaie, M. Schanz, F. Schreiber, and P. Levi

IPVS, University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany
robocup@informatik.uni-stuttgart.de

Abstract. The control software of the CoPS robot soccer team is designed as a Multi-Agent-System. The basis for cooperation is a suitable, flexible environment model which is synthesized by data processors based on uncertain information provided by sensors and communication.

1 Introduction

In the last four years, the CoPS-team (Cooperative Soccer Playing Robots) competed successfully in RoboCup tournaments. The control software of our robot team is designed as a Multi-Agent-System. Because of limitations of our former software structure we redesigned wide parts to provide a more general and maintainable software system, which can also be used outside the RoboCup-scenario.

2 System Architecture

The control software of our robots consists of several modules. Some for action selection and execution, some for sensing and some for modeling the environment. The architecture is shown in figure 1.

To get the information needed to select, plan, and execute an action we use a world model which is organized as a central repository in each robot. It holds all relevant data needed to control the behavior of the robot as a single agent and its cooperation within the team. Interrelations between the the world model and other modules are shown in figure 1.

The world model acts as a container for all knowledge of the robot. Two types of information are stored in the world model: sensory (raw) data and results of further processing. Therefor, a sensor module can register at the world model with its sensor type and feature type. A corresponding data object is created and can be accessed by the robot control modules. Also, it can be accessed by data processors, which provide more sophisticated algorithms. Because some algorithms, e.g. for tracking, need information about temporal state changes, a history of the data of the world model is provided.

The control part of the robot software has three major layers: the player, the navigator, and the pilot. The player's main task is the action selection for the individual robot and its team behavior, including response to external requests,

like referee decisions. The navigator executes the commands set by the player. It is responsible for finding the way to a given target point, e.g. by using a path planner, and reacting on moving obstacles during the movement. It also can execute complex orders, e.g. dribbling the ball or trying to score a goal. Therefor is can use a path-planner, with is also integrated in the control structure. This path-planner can easily be exchanged. The pilot is responsible for executing the navigator orders. It is the interface to the motor control board.

With this structure, we provide a universal structure for many applications in the area of mechatronic systems, not only RoboCup.

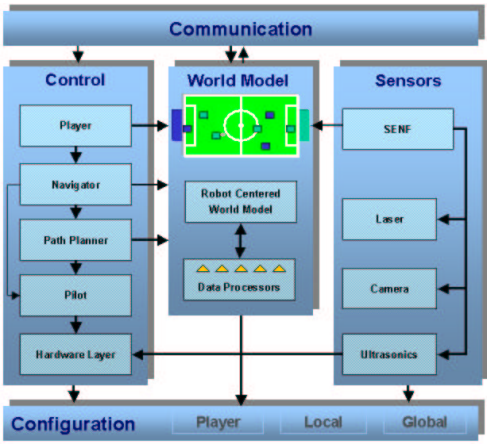

Fig. 1. Robot Software Architecture

3 Data Management and Selective Attention Control

One of the most important issues in the design of the control software for an autonomous robot is the handling and storing of information about the dynamic environment and the state of the robot. We use a "world model" as a central component to store all relevant information. The information to be stored consists of

- sensory (raw) data
- results of sensor data processing, fusing and merging (data driven)
- results of model driven procedures
- information communicated by other robots of the team
- control input, for RoboCup e.g. referee decisions.
- control parameters affecting the behavior of some modules or the robot

The basic data structures in our new implementation are data units. Specializations of this abstraction hold the actual data, e.g. easy integer arrays, together

with some general informations like timestamps. These data units are managed within so called data objects in a uniform way, using just the abstract interface. These data objects hold a list of data units of one feature and some administrative data. The list is used both for providing a short time history, for performance enhancements and managing reasons. The data objects are stored and managed within a data container. Such a container incorporates data from different sources with different semantics in a standardized format.

The described data objects act also as publishers according to the publisher-subscriber pattern. Data processors do some computation, like analysis, fusion and so on, and register at the data objects they require. This way they are informed each time these data objects change.

Data processors can be run either as separate threads or sequentialized in some common threads. Each data processor is configurable by means of a configuration object, which is also a data object as described above. Beside some special configuration data, each configuration object holds some information which controls the execution behavior, like the execution time one thread cycle should take. This allows to activate and deactivate or to "accelerate and slow down"[1] processors selectively.

Sensors are special data processors, normally running in separate threads. Using the configuration objects of the sensors, it is now possible to selectively effect the update cycles or the preprocessing of sensor data. The player module can now control the attention depending on the evaluated scenario.

Communication is handled using a CORBA event channel mechanism. Each robot registers itself at one or more event channels as consumer and/or supplier of information. Just the data objects which have to be communicated have to be registered at the communication module to be sent over the channel. Received data is stored in separate containers for each participating robot.

A more detailed description can be found in [?] and on our web pages.

Our new implementation allows to integrate new data processing modules into the system in a easy way, which enables fast prototyping and testing of new ideas. Also logging and debugging features are included in the base classes. To run some sensors and data processors in separate threads also introduces some uncertainty for example about the concrete processing order which made live-locks within the game less frequent.

4 Current Work

After the changeover to our new software structure we now concentrate on the following topics:

- New features for Monte-Carlo localization to make it more reliable.
- Integration of new sensors like omni-directional vision and a digital compass.
- plausibility analysis based on local world model information **and** communicated data.

[1] This means triggering the execution cycle of the data processor more or less often.

- Coordination of autonomous robots using coupled selection equations [?,?].
- Learning of single agent behaviors, which will be extended to learning for multi-agent-coordination.
- Taking full advantage of our new concept for selective attention control, for example by adding different data processors for the same task. Which task is executed is dynamically selected by the player component, depending on the accuracy or quickness of the algorithm.

5 Conclusion and Outlook

Based on our experiences in the RoboCup in the last years, we developed a new world model architecture. This structure now allows a faster and more efficient realization and integration of new software modules for data processing and robot control. Currently we are working on improvements for our existing data processors and we also evaluate several other concepts for their usability in the RoboCup scenario. On the hardware side, we are currently developing a new robot platform with omni-directional drive. One of the goals of the new software structure is easy migration to a completely different hardware with different sensors a minimum on changes on the higher levels.

References

[BBB+00] M. Becht, T. Buchheim, P. Burger, G. Hetzel, G. Kindermann, R. Lafrenz, N. Oswald, M. Schanz, M. Schulé, P. Molnár, J. Starke, and P. Levi. Three-index assignment of robots to targets: An experimental verification. In *IAS-6*, 2000.

[BKL03] T. Buchheim, G. Kindermann, and R. Lafrenz. A dynamic environment modelling framework for selective attention. In *IICAI Workshop: Issues in Designing Physical Agents for Dynamic Real-Time Environments: World modeling, planning, learning, and communicating.* IJCAI, 2003.

[SSF+02] M. Schulé, M. Schanz, H. Felger, J. Starke, and P. Levi. Dynamic control of autonomous robots using coupled selection equations. In *Robotik 2002*, 2002.