Decisive Factors for the Success of the Carologistics RoboCup Team in the RoboCup Logistics League 2014

Tim Niemueller[1(✉)], Sebastian Reuter[2], Daniel Ewert[2], Alexander Ferrein[3], Sabina Jeschke[2], and Gerhard Lakemeyer[1]

[1] Knowledge-based Systems Group, RWTH Aachen University, Aachen, Germany
niemueller@kbsg.rwth-aachen.de
[2] Institute Cluster IMA/ZLW and IfU, RWTH Aachen University, Aachen, Germany
[3] Electrical Engineering Department, FH Aachen, Aachen, Germany

Abstract. The RoboCup Logistics League is one of the youngest application- and industry-oriented leagues. Even so, the complexity and level of difficulty has increased over the years. We describe decisive technical and organizational aspects of our hardware and software systems and (human) team structure that made winning the RoboCup and German Open competitions possible in 2014.

1 Introduction

The Carologistics RoboCup Team is a cooperation of the Knowledge-based Systems Group, the IMA/ZLW & IfU Institute Cluster (both RWTH Aachen University), and the FH Aachen University of Applied Sciences initiated in 2012. Doctoral, master, and bachelor students of all three partners participate in the project and bring in their specific strengths tackling the various aspects of the RoboCup Logistics League sponsored by Festo (LLSF): designing hardware modifications, developing functional software components, system integration, and high-level control of a group of mobile robots.

Our approach to the league's challenges is based on a distributed system where robots are individual autonomous agents that coordinate themselves by communicating information about the environment as well as their intended actions. In this paper we outline decisive factors for our successes, like building on and extending proven methods and components and hosting events to attract new students to cope with the challenge of students leaving the team.

Our team has participated in RoboCup 2012, 2013, and 2014 and the RoboCup German Open (GO) 2013 and 2014. We were able to win the GO 2014 (cf. Fig. 1) as well as the RoboCup 2014 in particular demonstrating a flexible task coordination scheme, and robust collision avoidance and self-localization.

In the following Sect. 2 we give an overview of the LLSF. Then we describe our team's robots, software components (Sect. 3), and aspects of our task coordination and simulation (Sect. 4). We detail our involvement in the league's organization and outreach events in Sect. 5, before concluding in Sect. 6.

R.A.C. Bianchi et al. (Eds.): RoboCup 2014, LNAI 8992, pp. 155–167, 2015.
DOI: 10.1007/978-3-319-18615-3_13

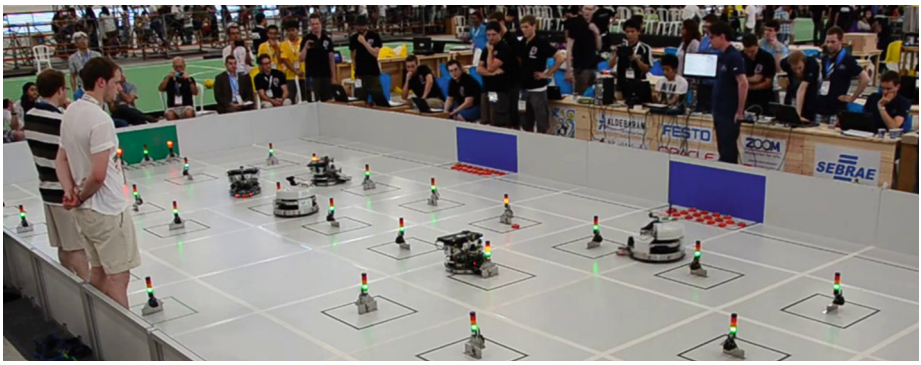


Fig. 1. Carologistics (three Robotino 2 with laptops on top) and BavarianBendingUnits (two larger Robotino 3) during the LLSF finals at RoboCup 2014 (Color figure online)

2 The RoboCup Logistics League

RoboCup [1] is an international initiative to foster research in the field of robotics and artificial intelligence. The basic idea of RoboCup is to set a common testbed for comparing research results in the robotics field. RoboCup is particularly well-known for its various soccer leagues. In the past few years focus application-oriented leagues such as urban search and rescue or domestic service robotics received more and more attention. In 2012, the new industry-oriented Logistics League Sponsored by Festo (LLSF) was founded to tackle the problem of production logistics. Groups of up to three robots have to plan, execute, and optimize the material flow in a factory automation scenario and deliver products according to dynamic orders. Therefore, the challenge consists of creating and adjusting a production plan and coordinate the group of robots [2]. The LLSF has attracted an increasing number of teams since established (8 teams in 2014).

The LLSF competition takes place on a field of 11.2 m × 5.6 m surrounded by walls (Fig. 1). Two teams are playing at the same time competing for points, (travel) space and time. Each team has an exclusive input storage (blue areas) and delivery zone (green area). Machines are represented by RFID-readers with signal lights on top. The lights indicate the current status of a machine, such as "ready", "producing" and "out-of-order". There are three delivery gates, one recycling machine, and twelve production machines per team. Material is represented by orange pucks with an RFID tag. At the beginning all pucks have the raw material state S_0 and are in the input storage, and can be refined through several stages to final products using the production machines. These machines are assigned a type randomly at the start of a match which determines what inputs are required and what output will be produced, and how long this conversion will take [3]. Finished products must then be taken to the active gate in the delivery zone.

The game is controlled by the referee box (refbox), a software component which keeps track of puck states, instructs the light signals, and posts orders to the teams [4]. After the game is started, no manual interference is allowed, robots receive instructions only from the refbox. Teams are awarded with points for delivering ordered products, producing complex products, and recycling. The refbox can be seen as a higher-level production planning entity as used in industry, e.g. ERP or MES-Systems.

3 The Carologistics Platform

The standard robot platform of this league is the Robotino by Festo Didactic [5]. The Robotino was developed for research and education and features omni-directional locomotion, a gyroscope and webcam, infrared distance sensors, and bumpers. It is also equipped with a static puck holder to move pucks. The teams may equip the robot with additional sensors and computation devices.

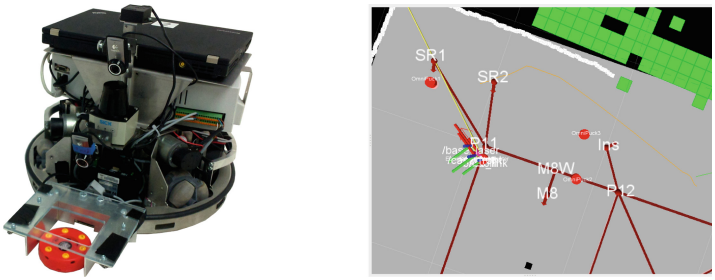
3.1 Hardware System

The robot system currently in use is still based on the Robotino 2 which is now replaced by the new version 3. The modified Robotino depicted in Fig. 2(a) used by the Carologistics RoboCup team features two additional webcams and a Sick laser range finder. One of the webcams is used for recognizing the signal lights of the production machines, the other to detect pucks in front of the robot. The former omni-directional camera is no longer used as it was prone to distortion and time-intensive calibration. Tracking pucks especially during rotational movement presented another challenge while the benefit of detecting pucks anywhere next to the robot was minimal. The webcams are mounted with serrated locking plates for a firm adjustment to defined angles. The Sick TiM551 laser scanner is used for collision avoidance and self-localization. In comparison to the Hokuyo laser scanner with a scanning range of 4 m we used last year, the Sick TiM551 has a maximal scanning of 10 m. An additional laptop on the robot increases the computation power and allows for more elaborate methods for self-localization, computer vision, and navigation. A custom-made passive guidance device is mounted to the front of the robots to allow for proper control of the pucks. Optical sensors mounted to the guidance device are used to measure the longitudinal distance for approaching the signal lights.

Next year we intend to migrate to the new Robotino 3. This is a requirement to cope with the new field stations coming in 2015 [4]. Preliminary experiments indicated a smooth migration of our control system to the new platform.

3.2 Architecture and Middleware

The software system of the Carologistics robots combines two different middlewares, Fawkes [6] and ROS [7]. This allows us to use software components from both systems. The overall system, however, is integrated using Fawkes.



(a) Carologistics Robotino 2014 (b) Visualization of a scene in rviz

Fig. 2. Carologistics Robotino, sensor processing, and visualization

Adapter plugins connect the systems, for example to use ROS' 3D visualization capabilities (cf. Fig. 2(b)). The overall software structure is inspired by the three-layer architecture paradigm [8]. It consists of a deliberative layer for high-level reasoning, a reactive execution layer for breaking down high-level commands and monitoring their execution, and a feedback control layer for hardware access and functional components. The lowest layer is described in Sect. 3.3. The upper two layers are detailed in Sect. 4. The communication between single components – implemented as *plugins* – is realized by a hybrid blackboard and messaging approach [6]. This allows for information exchange between arbitrary components. As shown in Fig. 3, information is written to or read from *interfaces*, each carrying certain information, e.g. sensor data or motor control, but also more abstract information like the position of an object. The information flow is somewhat restricted – by design – in so far as only one component can write to an interface. Reading, however, is possible for an arbitrary number of components. This approach has proven to avoid race conditions when for example different components try to instruct another component at the same time. The principle is that the interface is used by a component to provide state information. Instructions and commands are sent as messages. Then, multiple conflicting commands can be detected or they can be executed in sequence or in parallel, depending on the nature of the commands.

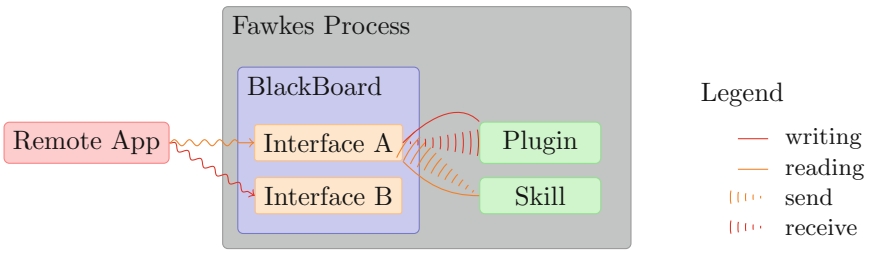


Fig. 3. Components communicate state data via interfaces stored in the blackboard. Commands and instructions are send as messages. Communication is universally shared among functional plugins and behavioral components.

3.3 Functional Software Components

A plethora of different software components is required for a multi-robot system. Here, we discuss the lowest layer in our architecture which contains functional modules and hardware drivers. All functional components are implemented in Fawkes. Drivers have been implemented based on publicly available protocol documentation, e.g. for our laser range finders or webcams. To access the Robotino base platform hardware we make use of a minimal subset of OpenRobotino, a software system provided by the manufacturer.

Localization is based on Adaptive Monte Carlo Localization which was ported from ROS and then extended. For locomotion, we integrated the collision avoidance module [9] which is also used by the AllemaniACs[1] RoboCup@Home robot.

A computer vision component for robust detection of the light signal state on the field has been developed specifically for this domain. A new such component we developed is a vision-based machine detection module. It will allow to detect and approach the machines more precisely as it yields a 3D pose. Figure 4 shows the visualization of the extracted features.

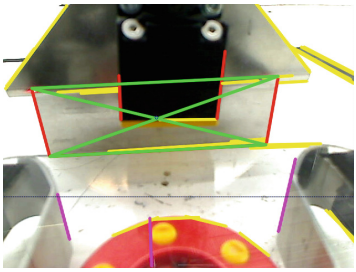


Fig. 4. Vision-based machine detection providing 3D pose information to approach a machine.

4 High-level Decision Making and Task Coordination

Task coordination is performed using an incremental reasoning approach [10]. In the following we describe the behavior components, and the reasoning process in two particular situations from the rules in 2014. For computational and energy efficiency, the behavior components need also to coordinate activation and deactivation of the lower level components to solve computing resource conflicts.

4.1 Behavior Components for the LLSF

Tasks that the high-level reasoning component of the robot must fulfill in the LLSF are:

Exploration: Gather information about the machine types by sensing and reasoning to gain more knowledge, e.g. the signal lights' response to certain types of pucks.

Production: Complete the production chains as often as possible dealing with incomplete knowledge.

Execution Monitoring: Instruct and monitor the reactive mid-level Lua-based Behavior Engine.

[1] See the AllemaniACs website at http://robocup.rwth-aachen.de.

A group of three robots perform these steps cooperatively, that is, they communicate information about their current intentions, acquire exclusive control over resources like machines, and share their beliefs about the current state of the environment. This continuous updating of information suggests an incremental reasoning approach. As facts become known, the robot needs to adjust its plan.

4.2 Lua-based Behavior Engine

In previous work we have developed the Lua-based Behavior Engine (BE) [11]. It mandates a separation of the behavior in three layers, as depicted in Fig. 5: the low-level processing for perception and actuation, a mid-level reactive layer, and a high-level reasoning layer. The layers are combined following an adapted hybrid deliberative-reactive coordination paradigm with the BE serving as the reactive layer to interface between the low- and high-level systems.

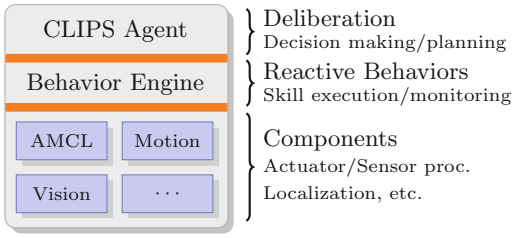


Fig. 5. Behavior Layer Separation

The BE is based on hybrid state machines (HSM). They can be depicted as a directed graph with nodes representing states for action execution, and/or monitoring of actuation, perception, and internal state. Edges denote jump conditions implemented as Boolean functions. For the active state of a state machine, all outgoing conditions are evaluated, typically at about 15 Hz. If a condition fires, the active state is changed to the target node of the edge. A table of variables holds information like the world model, for example storing numeric values for object positions. It remedies typical problems of state machines like fast growing number of states or variable data passing from one state to another. Skills are implemented using the light-weight, extensible scripting language Lua.

4.3 Incremental Reasoning Agent

The problem at hand with its intertwined world model updating and execution naturally lends itself to a representation as a fact base with update rules for triggering behavior for certain beliefs. We have chosen the CLIPS rules engine [12], because using incremental reasoning the robot can take the next best action at any point in time whenever the robot is idle. This avoids costly re-planning (as with approaches using classical planners) and it allows us to cope with incomplete knowledge about the world. Additionally, it is computationally inexpensive.

The CLIPS rules are roughly structured using a fact to denote the current overall state that determines which subset of the rules is applicable at any given time. For example, the robot can be idle and ready to start a new sub-task, or it may be busy moving to another location. Rules involved with physical interaction typically depend on this state, while world model belief updates often do not.

The state is also required to commit to a certain action and avoid switching to another one if new information, e.g., contributed by other robots on the field, becomes available. While it may be better in the current situation to pursue another goal, aborting or changing an action usually incurs much higher costs.

The rules explained in the following demonstrate what we mean by incremental reasoning. The robot does not create a full-edged plan at a certain point in time and then executes it until this fails. Rather, when idle it commits to the 'then-best' action. As soon as the action is completed and based on its knowledge, the next best action is chosen. The rule base is structured in six areas: exploration, production step decision, coordination with other robots, process execution, world modeling, and utilities.

```
(defrule load-T5-with-S0
  (declare (salience ?*PRIORITY-LOAD-T5-WITH-S0*))
  (phase PRODUCTION)
  ?s <- (state IDLE)
  (holding NONE|S0)
  (team-color ?team-color)
  (machine (mtype T5) (loaded-with $?l&~:(member$ S0 ?l))
    (incoming $?i&~:(member$ BRING_S0 ?i)) (name ?name)
    (produced-puck NONE) (team ?team-color))
  =>
  (printout t "PROD: Loading T5 " ?name " with S0" crlf)
  (assert (proposed-task (name load-with-S0) (args (create$ ?name))))
  (retract ?s)
  (assert (state TASK-PROPOSED))
)
```

Fig. 6. CLIPS production process rule

In Fig. 6 we show a simplified rule for the production process. The game is in the production phase, the robot is currently idle and holds a raw material puck S_0 or no puck: `(phase PRODUCTION) (state IDLE)(holding NONE|S0)`. Furthermore there is a $T5$-machine, whose team-color matches the team-color of the robot, which has no produced puck, is not already loaded with an S_0, and no other robot is currently bringing an S_0. If these conditions are satisfied and `*PRIORITY-LOAD-T5-WITH-S0*` is the highest priority of currently active rules, the rule fires proposing to load the machine with the name `?name` with an puck in state S_0. It also switches the state.

There is a set of such production rules with their conditions and priorities determining what the robot does in a certain situation, or – in other terms – based on a certain belief about the world in the fact base. This simplifies adding new decision rules. The decisions can be made more granular by adding rules with more restrictive conditions and a higher priority.

After a proposed task was chosen, the coordination rules of the agent cause communication with the other robots to announce the intention and ensure that there are no conflicts. If the coordination rules accept the proposed task, process execution rules perform the steps of the task (e.g. getting an S_0 from the input

storage and bringing it to the machine). Here, the agent calls the Behavior Engine to execute the actual skills like driving to the input storage and loading a puck.

The *world model* holds facts about the machines and their state, what kind of puck the robot is currently holding (if any) and the state of the robot. A simplified examples for a world model update is shown in Fig. 7. The world model update rule is invoked after a task or sub-task from the production rule presented above was successfully completed, i.e. an S_0 puck was taken to a machine of the type T_5. The rule shows the inference of the output puck type given a machine's reaction. The conditions `(state GOTO-FINAL) (goto-target ?name)` denote that the robot finished locomotion and production at the target machine `?name`, Furthermore, the robot sees only a green light at the machine, which indicates that the machine successfully finished the production. If all these conditions hold, the rule updates the world model about what kind of puck the robot is holding. Additionally it assumes all pucks removed that were loaded in the machine and increases the amount of consumed pucks. The world model is synchronized with other robots with another set of rules.

In comparison to 2013, the agent evolved to enable a tighter cooperation of the three agents. This required smaller atomic tasks, which are performed by the agents, a coordination mechanism to ensure the robots perform no redundant actions, more fine-grained production rules, and synchronization of the world model. The latter allows for dynamically adding or removing robots without interference to the overall production process. Furthermore, the agent became more robust against failure of behavior execution and wrong perception by adding a set of more distinctive world model update rules.

4.4 Multi-robot Simulation in Gazebo

The character of the LLSF game emphasizes research and application of methods for efficient planning, scheduling, and reasoning on the optimal work order of

```
(defrule goto-proc-complete
  (declare (salience ?*PRIORITY-WM*))
  (state GOTO-FINAL)
  (goto-target ?name)
  ?h <- (holding ?)
  (lights GREEN-ON YELLOW-OFF RED-OFF)
  (machine (name ?name) (output ?output) (loaded-with $?lw) (junk ?jn))
  =>
  (printout t "Production completed at " ?name "|" ?mtype crlf)
  (retract ?h)
  (assert (holding ?output))
  (foreach ?puck ?lw
    (assert (worldmodel-change (machine ?name)
              (change REMOVE_LOADED_WITH) (value ?puck)))
  )
  (assert (worldmodel-change (machine ?name) (change SET_NUM_CO)
                             (amount (+ ?jn (length$ ?lw)))))
)
```

Fig. 7. CLIPS world model update rules

production processes handled by a group of robots. An aspect that distinctly separates this league from others is that the environment itself acts as an agent by posting orders and controlling the machines' reactions. This is what we call *environment agency*. Naturally, dynamic scenarios for autonomous mobile robots are complex challenges in general, and in particular if multiple competing agents are involved. In the LLSF, the large playing field and material costs are prohibitive for teams to set up a complete scenario for testing, let alone to have two teams of robots. Additionally, members of related communities like planning and reasoning might not want to deal with the full software and system complexity. Still they often welcome relevant scenarios to test and present their research. Therefore, we have created an *open simulation environment* [3] to support research and development. There are three core aspects in this context: (1) The simulation should be a turn-key solution with simple interfaces, (2) the world must react as close to the real world as possible, including in particular the machine responses and signals, and (3) various levels of abstraction are desirable depending on the focus of the user, e.g. whether to simulate laser data to run a self-localization component or to simply provide the position (possibly with some noise) Fig. 8.

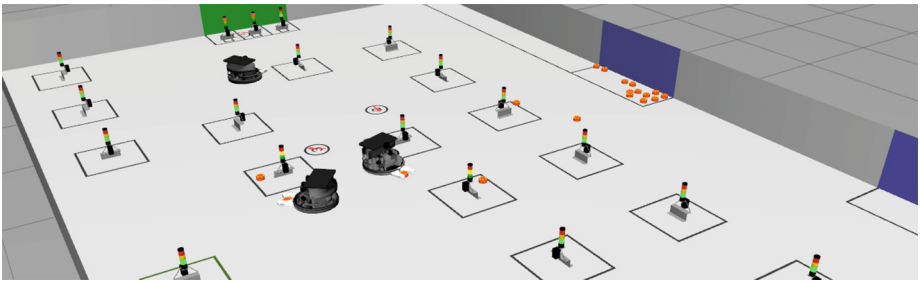


Fig. 8. The simulation of the LLSF in Gazebo. The circles above the robots indicate their localization and robot number.

In recent work [3], we provide such an environment. It is based on the well-known Gazebo simulator addressing these issues: (1) its wide-spread use and open interfaces already adapted to several software frameworks in combination with our models and adapters provides an easy to use solution; (2) we have connected the simulation directly to the referee box, the semi-autonomous game controller of the LLSF, so that it provides precisely the reactions and *environment agency* of a real-world game; (3) we have implemented *multi-level abstraction* that allows to run full-system tests including self-localization and perception or to focus on high-level control reducing uncertainties by replacing some lower-level components using simulator ground truth data. This allows to develop an idealized strategy first, and only then increase uncertainty and enforce robustness by failure detection and recovery. More information, media, and the software itself are available at http://www.fawkesrobotics.org/projects/llsf-sim/.

In the LLSF, the large playing field and material costs are prohibitive for teams to set up a complete scenario, let alone to have two teams of robots.

Additionally, members of related communities like planning and reasoning might not want to deal with the full software and system complexity. Still they often welcome relevant scenarios to test and present their research. Therefore, we propose a new simulation sub-league for the LLSF based on our simulation [3].

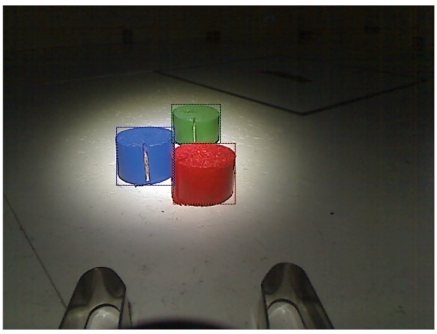
5 Continued Involvement and Outreach

We have been active members of the Technical and Organizational Committees and proposed various ground-breaking changes for the league like merging the two playing fields or using physical processing machines in 2015 [2,4]. Additionally we introduced and currently maintain the autonomous referee box for the competition as explained in the next section.

5.1 LLSF Referee Box

The Carologistics team has developed the autonomous referee box (refbox) for the LLSF which was deployed in 2013 [2]. It strives for full autonomy on the game controller, i.e. it tracks and monitors all puck and machine states, creates (randomized) game scenarios, handles communication with the robots, and interacts with a human referee. In 2014 the refbox has been adapted to the merged fields and two opposing teams on the field at the same time. We have also implemented a basic encryption scheme for secured communications.

(a) Field with illuminated obstacles where the robot has to find objects in the dark with just a headlight.



(b) View from the robot's front camera with detected pucks in the light cone of the headlight mounted on the robot.

Fig. 9. The 2014 Carologistics Hackathon challenge

5.2 Organizational and Didactic Aspects

One of the primary goals of the RoboCup Logistics League lies in providing a test bed for cyber-physical systems [13]. The competition also serves as an excellent

educational tool to give students a hands-on experience in dealing with robotics in industrial applications and future challenges for industry and research.

To improve the outreach of the league and involve a larger group of students the team and its associated institutes host *Hackathons*: students are invited to delve into the teams robotic system and develop a solution for a specific simplified challenge in a night's time. This year, the task was to explore an arena and find color-coded objects. A particular complication was that the arena was dark, i.e. without external lighting (see Fig. 9). The robots were equipped with headlights to enable perception with a considerably smaller field of view.

Hackathons also serves as recruiting platform for new team members, which are always needed due to the natural fluctuation, e.g. due to students leaving the university. Until now, Hackathons have been held each year since 2012, attracting up to 50 attendees for each event. The majority of the current student team members entered through one of the Hackathons.

The LLSF is also a formidable teaching platform. The KBSG offers regular lab courses where students are introduced to the robot platform and have to work on a specific problem. Example topics are inter-robot communication[2] or a new task coordination component[3] based on Procedural Reasoning Systems [14].

5.3 Research

The LLSF provides an excellent domain for research, in particular for a focus on task coordination for multi-robot systems. Compared to other RoboCup leagues, the problem is less dynamic (compared to soccer) and less cluttered and unorganized (compared to urban search and rescue or domestic service robots). That does not mean that it is easy to compete in the league but it does provide an interesting balance for researchers from related fields like planning or knowledge representation and reasoning to apply their results in a robotic environment.

In our context, creating a new central planning component is slated for inclusion as a part for the next phase of an ongoing research project on hybrid reasoning[4]. There, we want to explore the possibility to have a globally optimizing reasoner that offers suggestions for the robot group to maximize the overall achievable score with the given resources. This inclusion in larger scale research projects is crucial to advance the state of the art in a team or a league.

6 Conclusion

In this paper we have outlined several decisive factors for winning all competitions and technical challenges in 2014 without loosing a single game. It is important to have a joint team from different areas to cope with the large variety of challenges, from hardware modification to software integration. Our robust and proven base system that has been developed and used for and in other RoboCup

[2] http://kbsg.rwth-aachen.de/teaching/WS2012/LabRoCoCo.
[3] http://kbsg.rwth-aachen.de/teaching/WS2014/LabPRoGrAMR.
[4] http://www.hybrid-reasoning.org.

leagues allowed us to focus on the domain-specific challenges like flexible and efficient task coordination. In particular our focus on behavioral and multi-robot coordination components and the availability of our Gazebo-based simulation environment were crucial advantages. Even though the more elaborate approach meant a disadvantage through higher complexity for the simpler rules in 2012 and 2013, it was worthwhile since we were able to cope with the increasing level of difficulty of the league more quickly. Finally, our outreach program organizing large yearly Hackathons to attract and recruit new students is crucial to keep the team vivid and compensate leaving team members.

The website of the Carologistics RoboCup Team with further information and media can be found at http://www.carologistics.org.

Acknowledgments. The team members in 2014 are: Daniel Ewert, Alexander Ferrein, Sabina Jeschke, Nicolas Limpert, Gerhard Lakemeyer, Matthias Löbach, Randolph Maaßen, Victor Mataré, Tobias Neumann, Tim Niemueller, Florian Nolden, Sebastian Reuter, Johannes Rothe, and Frederik Zwilling.

We gratefully acknowledge the financial support of RWTH Aachen University and FH Aachen for participation at RoboCup 2014 in João Pessoa, Brazil. We thank the Bonding student organization for co-organizing and providing food, caffeinated drinks, and support for the Hackathons 2013 and 2014.

F. Zwilling and T. Niemueller were supported by the German National Science Foundation (DFG) research unit *FOR 1513* on Hybrid Reasoning for Intelligent Systems (http://www.hybrid-reasoning.org).

References

1. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E.: RoboCup: The Robot World Cup Initiative. In: Proceedings of the First International Conference on Autonomous Agents (1997)
2. Niemueller, T., Ewert, D., Reuter, S., Ferrein, A., Jeschke, S., Lakemeyer, G.: RoboCup logistics league sponsored by Festo: a competitive factory automation testbed. In: Behnke, S., Veloso, M., Visser, A., Xiong, R. (eds.) RoboCup 2013. LNCS, vol. 8371, pp. 336–347. Springer, Heidelberg (2014)
3. Zwilling, F., Niemueller, T., Lakemeyer, G.: Simulation for the RoboCup logistics league with real-world environment agency and multi-level abstraction. In: RoboCup Symposium (2014)
4. Niemueller, T., Lakemeyer, G., Ferrein, A., Reuter, S., Ewert, D., Jeschke, S., Pensky, D., Karras, U.: Proposal for advancements to the LLSF in 2014 and beyond. In: ICAR - 1st Workshop on Developments in RoboCup Leagues (2013)
5. Karras, U., Pensky, D., Rojas, O.: Mobile robotics in education and research of logistics. In: IROS 2011 - Workshop on Metrics and Methodologies for Autonomous Robot Teams in Logistics (2011)
6. Niemueller, T., Ferrein, A., Beck, D., Lakemeyer, G.: Design principles of the component-based robot software framework fawkes. In: Ando, N., Balakirsky, S., Hemker, T., Reggiani, M., von Stryk, O. (eds.) SIMPAR 2010. LNCS, vol. 6472, pp. 300–311. Springer, Heidelberg (2010)

7. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source Robot Operating System. In: ICRA Workshop on Open Source Software (2009)
8. Gat, E.: Three-layer Architectures. In: Kortenkamp, D., Bonasso, R.P., Murphy, R. (eds.) . Artificial Intelligence and Mobile Robots, pp. 195–210. MIT Press, Cambridge (1998)
9. Jacobs, S., Ferrein, A., Schiffer, S., Beck, D., Lakemeyer, G.: Robust collision avoidance in unknown domestic environments. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 116–127. Springer, Heidelberg (2010)
10. Niemueller, T., Lakemeyer, G., Ferrein, A.: Incremental task-level reasoning in a competitive factory automation scenario. In: Proceedings of AAAI Spring Symposium 2013 - Designing Intelligent Robots: Reintegrating AI (2013)
11. Niemüller, T., Ferrein, A., Lakemeyer, G.: A Lua-based behavior engine for controlling the humanoid robot nao. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 240–251. Springer, Heidelberg (2010)
12. Wygant, R.M.: A powerful development and delivery expert system tool. Comput. Ind. Eng. **17**(1–4), 546–549 (1989)
13. Niemueller, T., Ewert, D., Reuter, S., Karras, U., Ferrein, A., Jeschke, S., Lakemeyer, G.: Towards benchmarking cyber-physical systems in factory automation scenarios. In: Timm, I.J., Thimm, M. (eds.) KI 2013. LNCS, vol. 8077, pp. 296–299. Springer, Heidelberg (2013)
14. Alami, R., Chatila, R., Fleury, S., Ghallab, M., Ingrand, F.: An architecture for autonomy. Int. J. Robot. Res. **17**(4), 315–337 (1998)