The Carologistics RoboCup Logistics Team 2015

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, Germany
[2] Institute Cluster IMA/ZLW & IfU, RWTH Aachen University, Germany
[3] Electrical Engineering Department, FH Aachen, Germany

Abstract. The Carologistics team participates in the RoboCup Logistics League for the fourth year. As a medium complex domain balancing implementation effort and significance the league requires the development and integration of various software components. We outline the approach with an emphasis on the modifications required for the new game in 2015 in terms of custom hardware and software components. Additionally, we report on our efforts regarding the development of the league itself like releasing our full software stack as open source software.

The team members in 2015 are Daniel Ewert, Alexander Ferrein, Nicolas Limpert, Matthias Löbach, Randolph Maaßen, David Masternak, Victor Mataré, Tobias Neumann, Tim Niemueller, Sebastian Reuter, Johannes Rothe, and Frederik Zwilling.

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. The team was 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 (RCLL): designing hardware modifications, developing functional software components, system integration, and

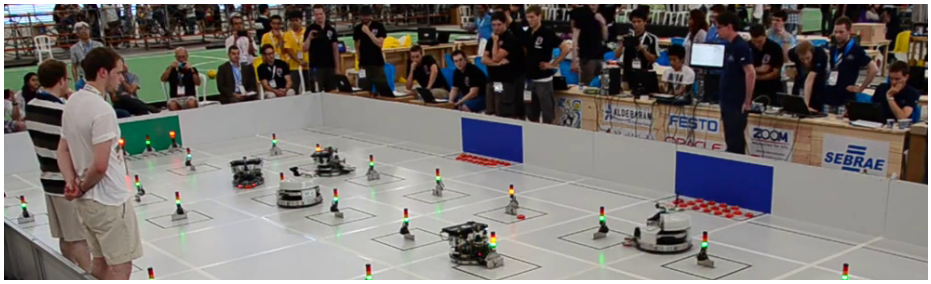

Fig. 1. Carologistics (three Robotino 2 with laptops on top) and BavarianBendingUnits (two Robotino 3) during the RCLL finals at RoboCup 2014 which ended 165 to 124.

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.

Our team has participated in RoboCup 2012–2014 and the RoboCup German Open (GO) 2013–2015. We were able to win the GO 2014 and 2015 (cf. Figure 1) as well as the RoboCup 2014 in particular demonstrating flexible task coordination, and robust collision avoidance and self-localization. We have publicly released our software stack used in 2014 in particular including our high-level reasoning components for all stages of the game[4] [1].

In the following we will describe some of the challenges originating from the new game play in 2015. In Section 2 we give an overview of the Carologistics platform and the changes that were necessary to adapt to the new game play. Some parts have been used during the German Open 2015, but several components were extended and improved afterwards. We continue highlighting our behavior components in Section 4 and our continued involvement for advancing the RCLL as a whole in Section 5 before concluding in Section 6.

1.1 Game Play Changes 2015

The goal is to maintain and optimize the material flow in a simplified Smart Factory scenario. Two competing groups of up to three robots each use a set of exclusive machines spread over a common playing field to produce and deliver products (cf. [2,3,4]).

In 2015, the RCLL has changed considerably by introducing actual processing machines based on the Modular Production System (MPS) platform by Festo Didactic [3] as shown in Figures 2 and 4. For more details on the new game play we refer to [5]. The new machines require to equip the robot with a gripper for product handling, adaptation to the general game play due to vastly extended production schedules, and suggest switching to the Robotino 3 platform, which is larger and supports a higher payload.

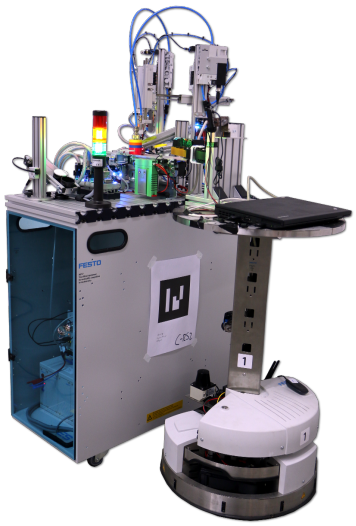

Fig. 2. Carologistics Robotino approaching a ring station MPS.

In the *exploration phase* the robots are given zones on the playing field in which they are supposed to look for machines, and – if one is found – identify and report them. In 2015, the machines can be freely positioned within the zones, in particular at any randomly chosen orientation. Additionally, the referee box can no longer provide ground truth information about the poses of machines in the production phase, making a successful exploration a mandatory requirement.

[4] Software stack available at `http://www.fawkesrobotics.org/p/llsf2014-release`

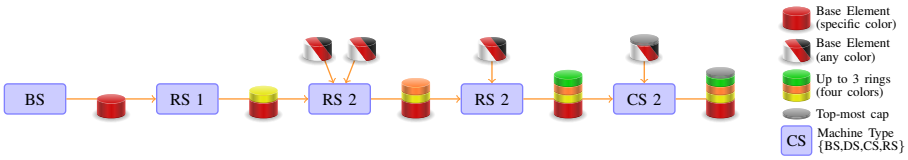

Fig. 3. Refinement steps for the production of a highest complexity product in the RCLL 2015 (legend on the right).

Additionally the exploration procedure per machine became more complex. The robots must first identify whether there is a machine in a zone or not, generally requiring multiple positions per zone to be checked. In our system, we sweep the zone with the laser scanner looking for machine edges and a camera looking for markers to make a quick decision whether a zone is empty or not. If a machine is identified, robots must align on the output side of a machine (frequently requiring to go around the machine which costs time) to identify the marker and light signal with high confidence.

In the *production phase*, work orders are much more diverse in 2015 increasing the number of product variants from 3 to about 250. An example production chain is shown in Figure 3. This in turn requires that the high-level reasoning component needs to make more decisions dynamically and opportunistic production is virtually impossible. It also requires to coordinate the robots more closely to achieve delivery in the desired time windows. Another challenge is that points are only awarded on successful delivery, even the points for intermediate processing steps. This requires the robots to be much more robust and the handling to be nearly perfect since harm done by failures in the production later along the chain is much more severe than before.

Fig. 4. Play-offs game at the German Open 2015

2 The Carologistics Platform

The standard robot platform of this league is the Robotino by Festo Didactic [6]. The Robotino is developed for research and education and features omnidirectional locomotion, a gyroscope and webcam, infrared distance sensors, and bumpers. The teams may equip the robot with additional sensors and computation devices as well as a gripper device for product handling.

2.1 Hardware System

The robot system currently in use is based on the Robotino 3. The modified Robotino used by the Carologistics RoboCup team is shown in Figure 5 and features three additional webcams and a Sick laser range finder. The webcam on the

top is used to recognize the signal lights, the one above the number to identify machine markers, and the one below the gripper is used experimentally to recognize the conveyor belt. We have recently upgraded to the Sick TiM571 laser scanner used for collision avoidance and self-localization. It has a scanning range of 25 m at a resolution of 1/3 degrees. An additional laptop increases the computation power and allows for more elaborate methods for self-localization, computer vision, and navigation.

Several parts were custom-made for our robot platform. Most notably, a custom-made gripper based on Festo fin-ray fingers and 3D-printed parts is used for product handling. The gripper is able to adjust for slight lateral offsets. We are currently investigating the possibility to also include a way of adjusting for height.

Fig. 5. Carologistics Robotino 2015

2.2 Architecture and Middleware

The software system of the Carologistics robots combines two different middlewares, Fawkes [7] and ROS [8]. This allows us to use software components from both systems. The overall system, however, is integrated using Fawkes. Adapter plugins connect the systems, for example to use ROS' 3D visualization capabilities. The overall software structure is inspired by the three-layer architecture paradigm [9]. 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 Section 3. The upper two layers are detailed in Section 4. The communication between single components – implemented as *plugins* – is realized by a hybrid blackboard and messaging approach [7]. This allows for information exchange between arbitrary components. As shown in Figure 6, 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.

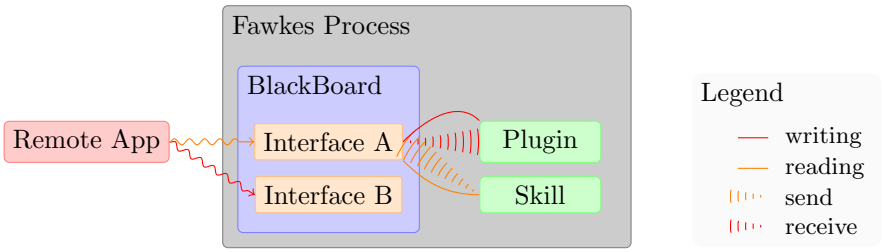

Fig. 6. Components communicate state data via interfaces stored in the blackboard. Commands and instructions are send as messages. Communication is universally shared among functional plugins and behavioral components.

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.

3 Advances to Functional Software Components

A plethora of different software components is required for a multi-robot system. Here, we discuss some components and advances of particular relevance to the game as played in 2015.

3.1 Basic Components

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. For this year, we have extended the driver of our laser range finder for the Sick TiM571 model. To access the Robotino base platform hardware we make use of a minimal subset of OpenRobotino, a software system provided by the manufacturer. For this year, we have upgraded to using the version 2 API in order to use the Robotino 3 platform. Localization is based on Adaptive Monte Carlo Localization which was ported from ROS and then extended. For locomotion, we integrated the collision avoidance module [10] which is also used by the AllemaniACs[5] RoboCup@Home robot.

3.2 Light Signal Vision

A computer vision component for robust detection of the light signal state on the field has been developed specifically for this domain. For 2015, we have improved the detection component to limit the search within the image by means of the detected position of the machine, which is recognized through a marker and with the laser range finder. This provides us with a higher robustness towards ambiguous backgrounds, for example colored T-shirts in the audience. Even if the machine cannot be detected, the vision features graceful degradation by using a geometric search heuristic to identify the signal, loosing some of the robustness towards the mentioned disturbances.

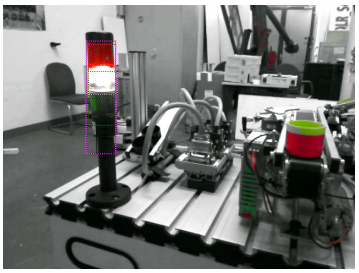

Fig. 7. Vision-based light-signal detection during production (post-processed for legibility).

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

3.3 Conveyor Belt Detection

The conveyor belts are rather narrow compared to the products thus require a precise handling. The tolerable error margin is in the range of about 3 mm. The marker on a machine allows to determine the lateral offset from the gripper to the conveyor belt. It gives a 3D pose of the marker with respect to the camera and thus the robot. However, this requires a precise calibration of the conveyor belt with respect to the marker. While ideally this would be the same for each machine, in practice there is an offset which would need to be calibrated per station. Combined with the inherent noise in the marker detection this approach requires filtering and longer accumulation times. The second approach is to detect a line using the laser scanner. This yields more precise information which we hope to further improve with the higher resolution of the Sick TiM571.

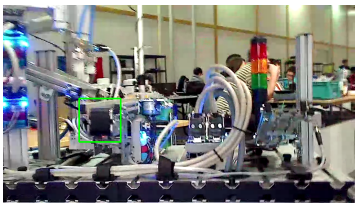

Fig. 8. Vision-based conveyor belt detection (training images).

The third approach we are investigating is using a dedicated vision component to detect the conveyor belt in an image. So far, we have implemented a detection method based on OpenCV's cascade classifier [11,12]. First, videos are recorded that are split into images from which positive and negative samples are extracted. These samples are then fed into a training procedure that extracts local binary features from which the actual classifiers are built. This training requires tremendous amounts of computing power and is generally done offline. The detection, however, is swift and allows for on-line real-time detection of the conveyor belt as shown in Figure 7. We are currently evaluating the results and considering further methods to improve on these results.

4 High-level Decision Making and Task Coordination

The behavior generating components are separated into three layers, as depicted in Figure 9: 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.

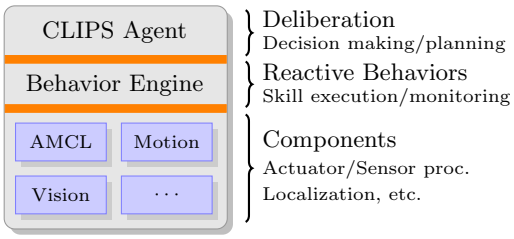

Fig. 9. Behavior Layer Separation

The robot group needs to cooperate on its tasks, that is, the robots communicate information about their current intentions, acquire exclusive control over resources like machines, and share their beliefs about the current state of the environment. Currently, we employ a distributed, local-scope, and incremental reasoning approach [5]. This means that each robot determines only its own action (local scope) to perform next (incremental) and coordinates with the others

through communication (distributed), as opposed to a central instance which plans globally for all robots at the same time or for multi-step plans.

In the following we describe the reactive and deliberative layers of the behavior components. For computational and energy efficiency, the behavior components need also to coordinate activation of the lower level components.

4.1 Lua-based Behavior Engine

In previous work we have developed the Lua-based Behavior Engine (BE) [13]. It serves as the reactive layer to interface between the low- and high-level systems. 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.

4.2 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 [14], 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. More details about the general agent design and the CLIPS engine are in [15].

The agent for 2015 is based on the previous years [15]. One major improvement is the introduction of a task concept. Until the 2014 version, the executive part of the agent consisted of separate rules for each action to perform. This resulted in a considerable amount of rules often repeating similar conditions. While this is not a problem in terms of performance, it does make maintaining the agent code more involved. For 2015, we have introduced the concept of tasks. Tasks group several steps necessary to perform a certain behavior, for example producing a low complexity product consisting only of a base element and a cap, into a single entity. Then, generic code can execute the steps of a task. We retained the flexibility and extensibility of our approach by allowing to add execution rules for steps which need special treatment or parametrization, and monitoring rules to react to disturbances during execution, for example a product being dropped during transport.

Additionally, we are progressing on making the world model synchronization generic. So far, we have had an explicit world model consisting of a specified set

of elements. Now, we are aiming for a more generic world model consisting of key-value pairs. This will allow for simpler updates to the world model as new additions, modifications, or deletions do not require changing an explicit world model schema anymore.

4.3 Multi-robot Simulation in Gazebo

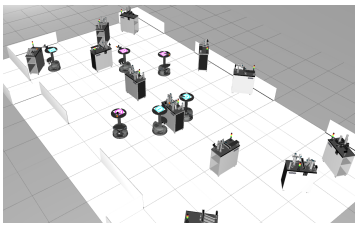

Fig. 10. Simulation of the RCLL 2015 with MPS stations.

The character of the RCLL game emphasizes research and application of methods for efficient planning, scheduling, and reasoning on the optimal work order of 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 RCLL, 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* [16,15] based on Gazebo.[6]

After RoboCup 2014, where the decisions to use MPS stations in 2015 and focus on the Robotino 3 as a platform have been made, the BBUnits team from TU Munich joined the effort and we cooperatively extended the simulation for the new game as shown in Figure 10. The simulation is developed publicly[7] and we hope for more teams to join the effort. With the new game, the need for a simulation is even more pressing as the cost for the field has drastically increased. Additionally, we envision that long-term games and future changes to the game can be tried in the simulation before implementing it in the real world [17].

5 League Advancements and Continued Involvement

We have been active members of the Technical, Organizational, and Executive Committees and proposed various ground-breaking changes for the league like merging the two playing fields or using physical processing machines in 2015 [2,3]. Additionally we introduced and currently maintain the autonomous referee box for the competition and develop the open simulation environment described above.

[6] More information, media, the software itself, and documentation are available at `http://www.fawkesrobotics.org/projects/llsf-sim/`

[7] The project is hosted at `https://github.com/robocup-logistics`

5.1 RCLL Referee Box and MPS Stations

The Carologistics team has developed the autonomous referee box (refbox) for the RCLL which was deployed in 2013 [2]. It strives for full autonomy on the game controller, i.e. it tracks and monitors 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.

In 2015, the refbox required updates for the modified game involving MPS stations. A noteworthy difference is that in the current game, the products can no longer be tracked as they do not have an RFID chip anymore. There are plans to remedy this later, for example using bar codes or RFID chips again on products. This would in particular allow to grant production points not only on delivery, but once a production step has been performed. The new MPS stations require programs that provide sensor data to and execute commands from the refbox. For example, all stations require a setup step to be performed, for example to instruct the particular ring color a ring station should mount in the next step. Festo Didactic sponsored an internship for a student who performed the development embedded in the Carologistics team. It involved developing a program for each of the four station types to reliably carry out the production steps and implementing a robust communication to the stations. As wifi is used for communication which is inherently unreliable in particular during RoboCup events, the refbox and stations must be robust towards temporary connection losses. The basic prototype was used successfully at the GO2015 and we are continuing the development to further stabilize and improve the game.

5.2 League Evaluation

We have made an effort [17] to analyze past games based on data recorded during games by the referee box and propose new additional criteria for game evaluation based on Key Performance Indicators (KPI) used in industrial contexts to evaluate factory performance. For example, one such KPI is the throughput time TTP which describes the time required at a specific station to complete processing and retrieve the processed workpiece. We have analyzed several games and will focus now on the finals of RoboCup 2014 between the Carologistics (cyan) and BBUnits (magenta) which ended with a score of 165 to 124.[8] It seems that especially the lower throughput time TTP of cyan contributed to the Carologistics' success. Machines can be used again much faster. This is shown by the green times in Figure 11, which is effective utilization of the machines. The BBUnits team has much longer waiting times (orange) that resulted for one in machines to be blocked for other uses and for another meant that the production chain for a product was not advanced quickly. For more details we refer to [17].

We imagine that KPIs can play a role in the future of the league and could be tested in simulated tournaments before. The Technical Committee has added a

[8] Video of the final is available at `https://youtu.be/_iesqH6bNsY`

technical challenge to the rules 2015 which involves performing such a simulated game in order to foster adoption and participation in the simulation efforts.

5.3 Public Release of Full Software Stack

Over the past eight years, we have developed the *Fawkes Robot Software Framework* [7] as a robust foundation to deal with the challenges of robotics applications in general, and in the context of RoboCup in particular. It has been developed and used in the Middle-Size [18] and Standard Platform [19] soccer leagues, the RoboCup@Home [20,21] service robot league, and now in the *RoboCup Logistics League* [15] as also shown in Figure 12.

The Carologistics are the first team in the RCLL to publicly release their software stack. Teams in other leagues have made similar releases before. What makes ours unique is that it provides a complete and *ready-to-run package with the full software* (and some additions and fixes) that we used in the competition in 2014. This in particular *includes* the complete *task-level executive* component,

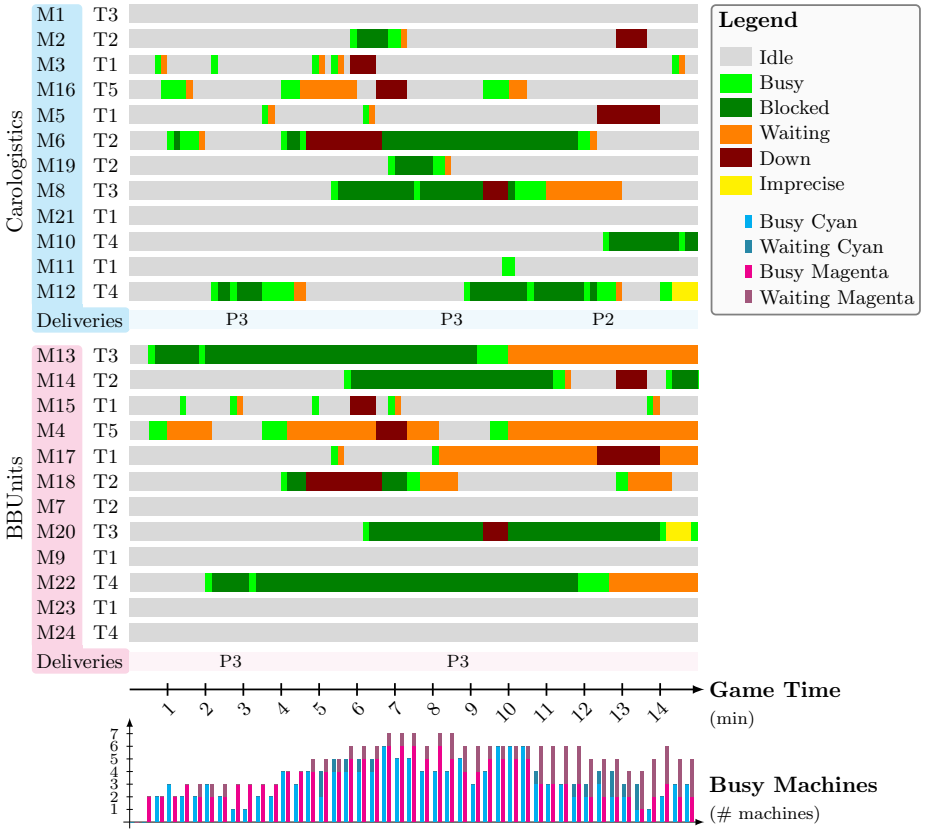

Fig. 11. Machine states over the course of the final game at RoboCup 2014. The lower graph shows the occupied machines per 20 sec time block.

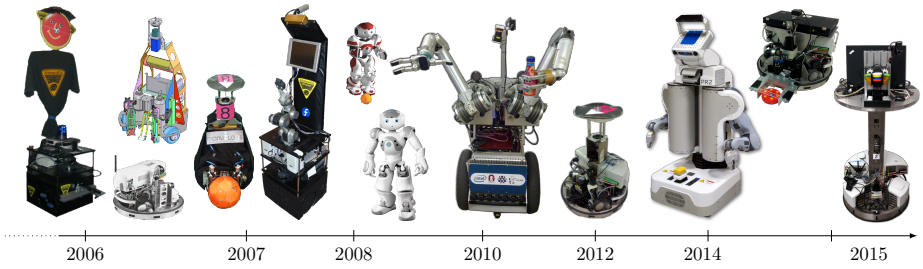

Fig. 12. Robots running (parts of) Fawkes which were or are used for the development of the framework and its components.

that is the strategic decision making and behavior generating software. This component was typically held back or only released in small parts in previous software releases by other teams (for any league).

6 Conclusion

In 2015, we have in particular adapted to the new game. We upgraded to the Robotino 3 platform, developed new custom hardware additions like a gripper, and adapted and extended the behavior and functional components. We have also continued our contributions to the league as a whole through active participation in the league's committees, publishing papers about the RCLL and proposing new performance criteria. The development of the simulation we initiated has been transferred to a public project where other teams have joined the effort. Most notably, however, have we released the complete software stack including all components and configurations as a ready-to-run package.

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

Acknowledgments. We gratefully acknowledge the financial support of RWTH Aachen University and FH Aachen University of Applied Sciences.
We thank Sick AG and Adolf Hast GmbH & Co. KG for sponsoring our efforts by providing hardware and manufacturing support.
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. Niemueller, T., Reuter, S., Ferrein, A.: Fawkes for the robocup logistics league. In: RoboCup Symposium 2015 – Development Track. (2015) to appear.
2. Niemueller, T., Ewert, D., Reuter, S., Ferrein, A., Jeschke, S., Lakemeyer, G.: RoboCup Logistics League Sponsored by Festo: A Competitive Factory Automation Benchmark. In: RoboCup Symposium 2013. (2013)

3. 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)
4. Niemueller, T., Lakemeyer, G., Ferrein, A.: Incremental Task-level Reasoning in a Competitive Factory Automation Scenario. In: Proc. of AAAI Spring Symposium 2013 - Designing Intelligent Robots: Reintegrating AI. (2013)
5. Niemueller, T., Lakemeyer, G., Ferrein, A.: The robocup logistics league as a benchmark for planning in robotics. In: 25th International Conference on Automated Planning and Scheduling (ICAPS) – Workshop on Planning in Robotics. (2015) to appear.
6. 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)
7. Niemueller, T., Ferrein, A., Beck, D., Lakemeyer, G.: Design Principles of the Component-Based Robot Software Framework Fawkes. In: Int. Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR). (2010)
8. 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)
9. Gat, E.: Three-layer architectures. In Kortenkamp, D., Bonasso, R.P., Murphy, R., eds.: Artificial Intelligence and Mobile Robots. MIT Press (1998) 195–210
10. 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 Symposium 2009. (2009)
11. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: IEEE Conf. on Comp. Vision and Pattern Recog. (CVPR). (2001)
12. Lienhart, R., Maydt, J.: An extended set of haar-like features for rapid object detection. In: Proc. Int. Conf on Image Processing. (2002)
13. Niemueller, T., Ferrein, A., Lakemeyer, G.: A Lua-based Behavior Engine for Controlling the Humanoid Robot Nao. In: RoboCup Symposium 2009. (2009)
14. Wygant, R.M.: CLIPS: A powerful development and delivery expert system tool. Computers & Industrial Engineering **17**(1–4) (1989)
15. Niemueller, T., Reuter, S., Ewert, D., Ferrein, A., Jeschke, S., Lakemeyer, G.: Decisive Factors for the Success of the Carologistics RoboCup Team in the RoboCup Logistics League 2014. In: RoboCup Symposium – Champion Teams Track. (2014)
16. 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)
17. Niemueller, T., Reuter, S., Ferrein, A., Jeschke, S., Lakemeyer, G.: Evaluation of the RoboCup Logistics League and Derived Criteria for Future Competitions. In: RoboCup Symposium 2015 – Development Track. (2015) to appear.
18. Beck, D., Niemueller, T.: AllemaniACs 2009 Team Description. Technical report, Knowledge-based Systems Group, RWTH Aachen University (2009)
19. Ferrein, A., Steinbauer, G., McPhillips, G., Niemueller, T., Potgieter, A.: Team ZaDeAt 2009 – Team Report. Technical report, RWTH Aachen Univ., Graz Univ. of Technology, and Univ. of Cape Town (2009)
20. Schiffer, S., Lakemeyer, G.: AllemaniACs Team Description RoboCup@Home. TDP, Knowledge-based Systems Group, RWTH Aachen University (2011)
21. Ferrein, A., Niemueller, T., Schiffer, S., and, G.L.: Lessons Learnt from Developing the Embodied AI Platform Caesar for Domestic Service Robotics. In: AAAI Spring Symposium 2013 - Designing Intelligent Robots: Reintegrating AI. (2013)