International Harting Open Source Award 2016: Fawkes for the RoboCup Logistics League

Tim Niemueller[1(✉)], Tobias Neumann[2], Christoph Henke[3], Sebastian Schönitz[3], Sebastian Reuter[3], Alexander Ferrein[2], Sabina Jeschke[3], and Gerhard Lakemeyer[1]

[1] Knowledge-based Systems Group, RWTH Aachen University, Aachen, Germany
niemueller@kbsg.rwth-aachen.de
[2] MASCOR Institute, FH Aachen University of Applied Sciences, Aachen, Germany
[3] Institute Cluster IMA/ZLW & IfU, RWTH Aachen University, Aachen, Germany

Abstract. Since 2014, we have made three releases of our full software stack for the *RoboCup Logistics League* (RCLL) based on the Open Source *Fawkes Robot Software Framework*. They include all software components of the team Carologistics which won RoboCup 2014, 2015, and 2016. The software is based on experience from participating in a number of leagues with the AllemaniACs RoboCup@Home team being another active contributor. We think that these releases have made the RCLL more accessible to new teams and helped established ones to improve their performance. The team is proud to have been selected for the third place of the 1st International Harting Open Source Award in 2016. In this paper, we give an overview of the framework and its development.

1 Introduction

Autonomous mobile robots comprise a great deal of complexity. They require a plethora of software components for perception, actuation, task-level reasoning, and communication. These components have to be integrated into a coherent and robust system in time for the next RoboCup event. Then, during the competition, the system has to perform stable and reliably. Providing a software framework for teams to use tremendously eases that effort. Even more so when providing a fully integrated system specific for a particular domain.

Over the past ten years, we have developed the *Fawkes Robot Software Framework* [2] 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 [3] and Standard Platform [4] soccer leagues, the RoboCup@Home [5,6] service robot league, and now in the *RoboCup Logistics League* [7,8]. The frameworks or parts of it have also been used in other contexts [9,10]. In Fig. 1 the timeline of some robots used with Fawkes is depicted. Although Fawkes is designed as a general framework to fit various robotics applications, in this paper we focus on its use in the RCLL.

S. Behnke et al. (Eds.): RoboCup 2016, LNAI 9776, pp. 634–642, 2017.
https://doi.org/10.1007/978-3-319-68792-6_53

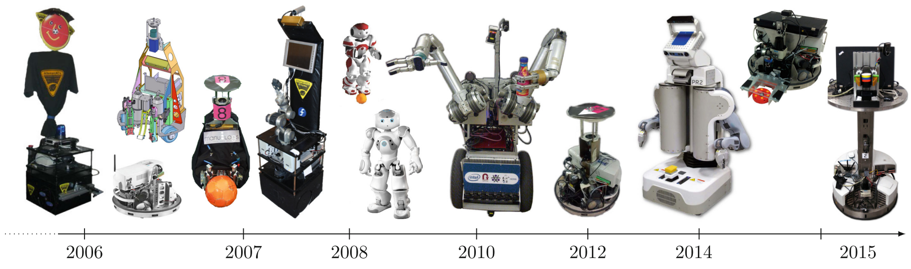

Fig. 1. Robots running (parts of) Fawkes which were or are used for the development of the framework and its components in the past ten years [1].

We have been the first team in the RCLL to publicly release their software stack. Teams in other leagues have made similar releases before [11]. 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 several competitions – which we won. This in particular *includes* the complete *task-level executive* component, that is the strategic decision making and behavior generating software. The major parts of the domain model are also made publicly available.

In the RCLL all teams use the same hardware platform "Robotino" by Festo Didactic. This means that there is *no hardware barrier* that prevents teams from using the software effectively and quickly. Even more so, with the *3D simulation environment* based on Gazebo which we have developed [12] and provide, teams can immediately start using our software system for their own development. We provide extensive documentation and are expanding it continuously.

In 2016, the RCLL software stack based on Fawkes[1] was selected for the third place of the 1st International Harting Open Source Prize.

In the following we will briefly describe the framework, some major components, and our simulation environment in Sect. 2 with a highlight on the task-level executive in Sect. 3. We conclude in Sect. 4.

2 Fawkes Robot Software Framework

The software stack is based on the *Fawkes Robot Software Framework*[2] which is Open Source software. The development is split into a core and domain-specific parts. The core framework, Fawkes, is developed in public. We have just released the first stable release 1.0. The domain-specific components are developed in private as they are considered to be our competitive edge. We have made several releases in the past few years, one after each RoboCup event since 2014.

Fawkes was initially started in 2006 as an effort to build a capable and faster software platform for a new generation of Mid-Size league robots of the *AllemaniACs*[3] *RoboCup Team* (cf. Fig. 1). It was used for the first time at RoboCup

[1] Latest release: https://www.fawkesrobotics.org/p/rcll2016-release.
[2] Fawkes website at https://www.fawkesrobotics.org.
[3] Website of the AllemaniACs at https://robocup.rwth-aachen.de.

2007 in Atlanta. Since then it was also used on our domestic service robot Caesar [6] in the RoboCup@Home league winning the RoboCup in 2006 and 2007, placing second in 2008, and winning the German Open 2007 and 2008 [5]. From 2008 to 2010 we participated as team ZaDeAt [4], a joint team from University of Cape Town (ZA), RWTH Aachen University (DE) and Technical University of Graz (AT), in the Standard Platform League. During this time we developed the Lua-based Behavior Engine [13], a component which was ported to ROS in 2010 and used, for example, on HERB at CMU [9]. Since 2012 we participate in the RoboCup Logistics League as the *Carologistics*[4] *joint team* consisting of the Knowledge-Based Systems Group, the Institute Cluster IMA/ZLW & IfU (both RWTH Aachen University), and the Institute for Mobile Autonomous Systems and Cognitive Robotics (FH Aachen University of Applied Sciences). We won the RoboCup and RoboCup German Open titles 2014–2016. Fawkes is also used in combination with ROS on a PR2 in a project on hybrid reasoning [10].

The overall software structure is designed as a three-layer architecture [14] and follows a component-based paradigm [15–17]. 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 communication between single components – implemented as *plugins* – is realized by a hybrid blackboard and messaging approach [2]. Other teams use monolithic approaches or messaging by standardized interfaces [18].

Fawkes and ROS

The most popular robot software framework is the Robot Operating System (ROS) [19]. It has a rich ecosystem of existing software components. Its development started at about the same time. Fawkes and ROS can be fully integrated, for example with Fawkes running as a ROS node. Some plugins have been extended directly to interact with ROS, e.g., for visualizing component-specific information, the main purpose of ROS on the Carologistics' and AllemaniACs' robots. Generic adapter plugins translate between the middleware differences and message types. For example, Fawkes can either provide its navigation capabilities to ROS, or integrate ROS' move_base component for path planning.

Fawkes uses a monolithic approach, running most components as dynamically loaded plugins multi-threaded in a single process, while ROS focuses on a multi-process approach of federated nodes. Fawkes uses a hybrid blackboard/messaging communication architecture, while ROS uses a publisher/subscriber middleware. While Fawkes uses a development model focused on a few core repositories used to develop the components, for ROS components are developed rather separately.

Software Components

Fawkes already contains a wide variety of more than 125 software components and more than two dozen software libraries, many of which are used in the RCLL.

[4] Website of the Carologistics at https://www.carologistics.org.

These cover a wide range of functionalities, from plugins providing infrastructure, over functional components for self-localization and navigation, and perception modules via point clouds, laser range finders, or computer vision, to behavior generating components following reactive or deliberative paradigms. In the following we describe some examples with a particular focus on the RCLL. The behavior components are explained in more detail in Sect. 3.

Navigation. Fawkes comes with an implementation of Adaptive Monte Carlo Localization which is an extended port from ROS. In the RCLL, we use a pre-specified map and a laser range finder to determine and track the position of the robot on the field. For locomotion path planning we use a layered structure. A component called *navgraph* has a topological graph of the playing field, where nodes specify travel points or points of interest like machines, and edges denote passages free from static obstacles. When moving to a specific point the navgraph plugin determines a path on this graph to reach the goal. It then instructs the *colli* [20], a local path planner and collision avoidance module we have developed. Based on the next (intermediate) goal on the path it follows a collision-free path.

Perception. The detection and recognition of the light signal of a machine as shown in Fig. 2. While it might seem like a routine task for computer vision, it is complicated by several factors. Since the lights can be on and off, the brightness of the image varies significantly. Additionally, background clutter colored alike the light signal makes detection difficult. A full search for the light signal in an image therefore results in many false positives and negatives. Thus we use a multi-modal laser-based search space reduction [21].

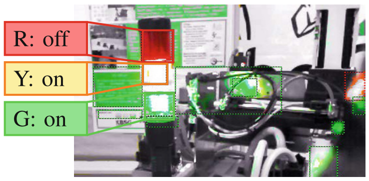

Fig. 2. Machine signal detection used in the RCLL 2016. The markings denote the detected lights [21].

Simulation. The RCLL 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. This is what we call *environment agency.* Therefore, we have created an *open simulation environment* [12] depicted in Fig. 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.

In recent work [12], we provide such an environment.[5] It is based on the well-known Gazebo simulator addressing these issues: (1) its wide-spread use

[5] Simulation is available at https://www.fawkesrobotics.org/p/rcll-sim/.

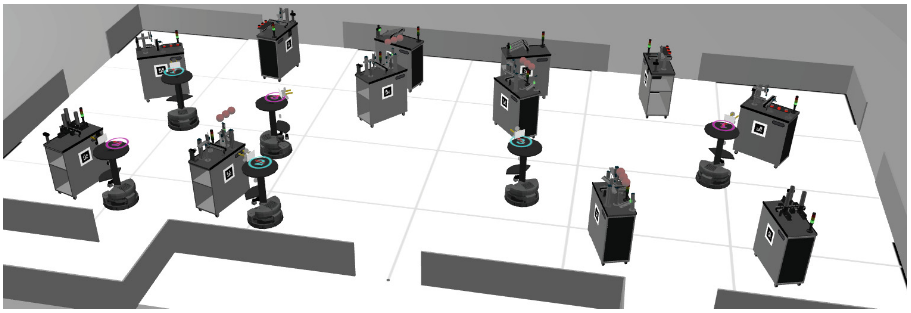

Fig. 3. The simulation of the RCLL in Gazebo based on Fawkes.

and open interfaces already adapted to several software frameworks in combination with our models and adapters provide an easy to use solution; (2) we have connected the simulation directly to the referee box, the semi-autonomous game controller of the RCLL, 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.

The simulation also forms the basis for a new logistics robots competition in simulation [22]. It is intended to build a bridge between the planning and robotics communities and foster closer cooperation for integrating state-of-the-art planning systems into a robotics scenario.

3 Task-Level Coordination and Execution

In the model as depicted in Fig. 4, behavior specification takes place in the upper two layers. The layers are combined following an adapted hybrid deliberative-reactive coordination paradigm. On the lower level, processing for perception and actuation takes place. Task coordination is performed using an incremental reasoning approach [23] on the top level and a reactive middle layers creates a consistent and unified interface to the lower level components. In the RCLL, the top level takes care about selecting the next tasks to accomplish and to coordinate with the other robots. The middle layer provides a reactive framework for modeling, implementing, executing, monitoring, and (locally repairing) basic skills like moving a place, but also multi-step actions like retrieving a workpiece.

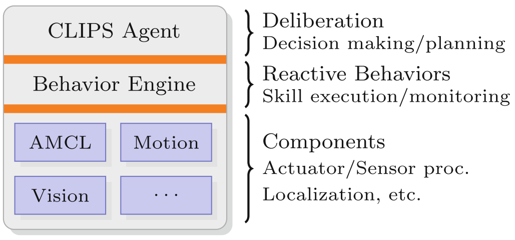

Fig. 4. Behavior layer separation [23].

For computational and energy efficiency, the behavior components need also to coordinate activation of the lower level components to solve computing resource conflicts.

In the following, we describe these two components as a core contribution of the Fawkes framework in the RCLL in a little more detail.

Lua-based Behavior Engine

In previous work we have developed the Lua-based Behavior Engine (BE) [13]. It integrates as a plugin into Fawkes and has also been ported to and used in ROS [9]. The ROS integration is also available as part of Fawkes allowing for a direct hybrid development of behaviors based on Fawkes and ROS.

The BE implements individual behaviors – called skills – as hybrid state machines (HSM). They can be depicted as a directed graph (cf. Fig. 5 to the right) with nodes representing states for action execution and monitoring. 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.

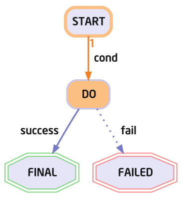

Fig. 5. Hybrid state machine.

For the RCLL, more than thirty skills have been implemented with a hierarchical structure where more complex skills like retrieving a workpiece build on more basic ones like approaching and aligning at an MPS.

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 [24]. *Incremental reasoning* means that the robot does not create a full-edged plan at a certain point in time and then executes it. Rather, when idle it commits to the “then-best” action. This avoids costly re-planning (as with approaches using planners), it allows to cope with incomplete knowledge about the world, and it is computationally inexpensive. The decision is based on the current situation as determined through a world model that is weakly synchronized with the other robots and eventually consistent [25]. Adding a new rule is simplified through more specific rules augmenting more general ones.

The robots must communicate to coordinate with the group in order to avoid multiple robots choosing the same task. A mechanism for mutual exclusion denotes one robot as leader through dynamic election. For each task to perform and resource to use, locks must be acquired ensuring that conflicts are resolved early. Robots who fail to obtain re-evaluate their choice with respect to the updated knowledge (that another robot is already performing that task).

Another set of rules controls and monitors the execution of the basic behaviors through the Behavior Engine to accomplish the task. For example, consider a task to retrieve a basic element and delivering it to another machine. This is broken down in several skills. Should the basic element be dropped on the way, the robot can repair the task by retrieving another one, or make a new decision.

4 Conclusion

The integration of a complete robot system even for medium-complex domains such as the RCLL can be tedious and time consuming. We had made the decision early in 2012 when joining the RCLL to go for a more complex, but then also more robust and flexible system. This was finally rewarded by winning the RoboCup 2014, 2015, and 2016 RCLL competitions.

The public release of a fully working and thoroughly tested integrated software stack lowers the barrier of entry for new teams to the league and fosters research and exchange among members of the RoboCup community in general, and in the RoboCup Logistics League in particular. We have organized the first RCLL Winter School in 2015 to disseminate this work and to discuss future directions with other members of the community. These effort were honored with the third place of the 1st International Harting Open Source Award 2016. We continue to develop Fawkes as Open Source software.

Acknowledgments. The Carologistics team members in 2015/2016 are: A. Ferrein, M. Gomaa, C. Henke, S. Jeschke, N. Limpert, D. Kuenster, G. Lakemeyer, M. Löbach, V. Mataré, T. Neumann, T. Niemueller, S. Reuter, J. Rothe, D. Schmidt, S. Schönitz, and F. Zwilling.

The AllemaniACs team members in 2015/2016 are: G. Gierse, T. Hofmann, B. Maleki-Fard, T. Niemueller, S. Schiffer, and F. Zwilling.

We gratefully acknowledge the financial support of RWTH Aachen University and FH Aachen University of Applied Sciences.

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

References

1. Niemueller, T., Reuter, S., Ferrein, A.: Fawkes for the RoboCup logistics league. In: Almeida, L., Ji, J., Steinbauer, G., Luke, S. (eds.) RoboCup 2015. LNCS, vol. 9513, pp. 365–373. Springer, Cham (2015). doi:10.1007/978-3-319-29339-4_31
2. 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). doi:10.1007/978-3-642-17319-6_29
3. Beck, D., Niemueller, T.: AllemaniACs 2009 Team Description. Technical report, KBSG, RWTH Aachen University (2009)

4. Ferrein, A., Steinbauer, G., McPhillips, G., Niemueller, T., Potgieter, A.: Team ZaDeAt 2009 - Team Report. Graz University of Technology, and University of Cape Town, Technical report, RWTH Aachen University (2009)
5. Schiffer, S., Lakemeyer, G.: AllemaniACs Team Description RoboCup@Home. Technical report, KBSG, RWTH Aachen University (2011)
6. Ferrein, A., Niemueller, T., Schiffer, S., Lakemeyer, G.: Lessons learnt from developing the embodied AI platform caesar for domestic service robotics. In: Proceedings of AAAI Spring Symp, 2013 - Designing Intelligent Robots: Reintegrating AI (2013)
7. 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: Bianchi, R.A.C., Akin, H.L., Ramamoorthy, S., Sugiura, K. (eds.) RoboCup 2014. LNCS, vol. 8992, pp. 155–167. Springer, Cham (2015). doi:10.1007/978-3-319-18615-3_13
8. Niemueller, T., Reuter, S., Ewert, D., Ferrein, A., Jeschke, S., Lakemeyer, G.: The carologistics approach to cope with the increased complexity and new challenges of the RoboCup logistics league 2015. In: Jeschke, S., Isenhardt, I., Hees, F., Henning, K. (eds.) Automation, Communication and Cybernetics in Science and Engineering 2015/2016, pp. 619–635. Springer, Cham (2016). doi:10.1007/978-3-319-42620-4_46
9. Srinivasa, S.S., Berenson, D., Cakmak, M., Collet, A., Dogar, M.R., Dragan, A.D., Knepper, R.A., Niemueller, T., Strabala, K., Vande Weghe, M., Ziegler, J.: HERB 2.0: lessons learned from developing a mobile manipulator for the home. Proc. IEEE **100**(8), 2410–2428 (2012)
10. Niemueller, T., Abdo, N., Hertle, A., Lakemeyer, G., Burgard, W., Nebel, B.: Towards deliberative active perception using persistent memory. In: Proceedings of the Workshop on AI-based Robotics at the International Conference on Intelligent Robots and Systems (IROS) (2013)
11. Röfer, T., Laue, T.: On B-Human's code releases in the standard platform league – software architecture and impact. In: Behnke, S., Veloso, M., Visser, A., Xiong, R. (eds.) RoboCup 2013. LNCS, vol. 8371, pp. 648–655. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44468-9_61
12. Zwilling, F., Niemueller, T., Lakemeyer, G.: Simulation for the RoboCup Logistics League with real-world environment agency and multi-level abstraction. In: Bianchi, R.A.C., Akin, H.L., Ramamoorthy, S., Sugiura, K. (eds.) RoboCup 2014. LNCS, vol. 8992, pp. 220–232. Springer, Cham (2015). doi:10.1007/978-3-319-18615-3_18
13. 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). doi:10.1007/978-3-642-11876-0_21
14. Gat, E.: Three-layer architectures. In: Artificial Intelligence and Mobile Robots. MIT Press (1998)
15. McIlroy, M.D.: Mass produced software components. In: Software Engineering: Report On a Conference Sponsored by the NATO Science Committee (1968)
16. Brugali, D., Scandurra, P.: Component-based robotic engineering (part I). IEEE Robot. Autom. Mag. **16**(4), 84–96 (2009)
17. Brugali, D., Shakhimardanov, A.: Component-based robotic engineering (part II). IEEE Robot. Autom. Mag. **17**(1), 100–112 (2012)
18. Mamantov, E., Silver, W., Dawson, W., Chown, E.: RoboGrams: a lightweight message passing architecture for RoboCup soccer. In: Bianchi, R.A.C., Akin, H.L., Ramamoorthy, S., Sugiura, K. (eds.) RoboCup 2014. LNCS, vol. 8992, pp. 306–317. Springer, Cham (2015). doi:10.1007/978-3-319-18615-3_25

19. 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)
20. 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). doi:10.1007/978-3-642-11876-0_11
21. Mataré, V., Niemueller, T., Lakemeyer, G.: Robust multi-modal detection of industrial signal light towers. In: RoboCup Symposium (2016, to appear)
22. Niemueller, T., Karpas, E., Vaquero, T., Timmons, E.: Planning competition for logistics robots in simulation. In: WS on Planning and Robotics (PlanRob) at International Conference on Automated Planning and Scheduling (ICAPS), London, UK (2016)
23. 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)
24. Wygant, R.M.: CLIPS: a powerful development and delivery expert system tool. Comput. Industr. Eng. **17**(1–4), 546–549 (1989)
25. Vogels, W.: Eventually Consistent. ACM Queue **6**(6), 14–19 (2008)