Elektrotechnik & Informationstechnik (2020) 137/6: 291–296. https://doi.org/10.1007/s00502-020-00826-5

A multi-robot architecture for the RoboCup Logistics League

P. Kohout, M. De Bortoli, J. Ludwiger, T. Ulz, G. Steinbauer

Due to increasing demands on flexibility in terms of product configuration as well as delivery time production settings increasingly use teams of mobile robot systems. The RoboCup Logistics League was designed to provide a testbed to develop and test such flexibe multi-robot approaches for production environments. It resembles a product setting with on-demand product orders of different configurations. In this article we introduce the concept of the leagues and we present the solution of the team GRIPS to face that challenge.

Keywords: flexible production; logistics; RoboCup; planning

Eine Multi-Roboter-Architektur für die RoboCup Logistics League.

Durch die erhöhten Erfordernisse an die Flexibilität der Produktkonfiguration sowie der Lieferzeiten verwenden Produktionsumgebungen vermehrt mobile Robotersysteme. Um eine Testumgebung für Entwicklung und Evaluierung solcher flexiblen Multi-Robot Ansätze für Produktionsumgebungen bereitzustellen, wurde die RoboCup Logistics League entworfen. Sie stellt eine Produktionsumgebung mit On-Demand-Bestellungen und individuellen Produktkonfigurationen dar. In diesem Artikel stellen wir das Konzept der Testumgebung sowie die Lösung für diese Herausforderung des Team GRIPS vor.

Schlüsselwörter: flexible Produktion; Logistik; RoboCup; Planung

Received May 29, 2020, accepted September 1, 2020, published online September 9, 2020

1. Introduction

Due to increasing demands on flexibility in terms of product configuration as well as delivery time triggered by the boom in e-commerce (e.g. on-line configurators, on-line shopping) production needs to become more flexible. This trend is well known under terms like flexible production or Industry 4.0. Usually in order to allow reasonable prices for products and to guarantee stable product quality and fast availability production is heavily automatized. The increasing demand for flexibility, in contrast with common rigidity of automation, asks for new concepts and opens interesting and challenging research questions ranging from Robotics over IoT and multi-agent systems to planning and scheduling. In order to provide an appealing show case that allows research and teaching in the area of flexible production within the RoboCup initiative [10] a competition called the RoboCup Logistics League (RCLL) was founded. It resembles the setting of a flexible production plant. In this paper we like to introduce the RCLL, to show the challenges that are posted by the competition, and describe how the competition can be used to develop and evaluate new concepts in production [11].

2. Logistics League

The RCLL [6, 8] is part of the RoboCup initiative and focuses on the stimulation of the development of approaches in Robotics and Artificial Intelligence using robotics competition. In this league the goal is that a team of robots in cooperation with a set of production machines produces products on demand. Two teams share a common factory floor of the size of 14 m × 8 m. Each team comprises of up to 3 autonomous robots and owns 7 machines, represented by Modular Production Systems (MPS) provided by Festo. See Fig. 1. There are different types of machines that resemble different production steps like fetching raw material, assembling parts, or delivering final products. The task of the teams is to develop methods that coordinate mobile robots and static machines. Robots and machines are allowed to communicate using WiFi. Robots are cooperative in the sense that they need to interact physically with the machines, e.g. fetching raw material from a dispenser machine or provide in-

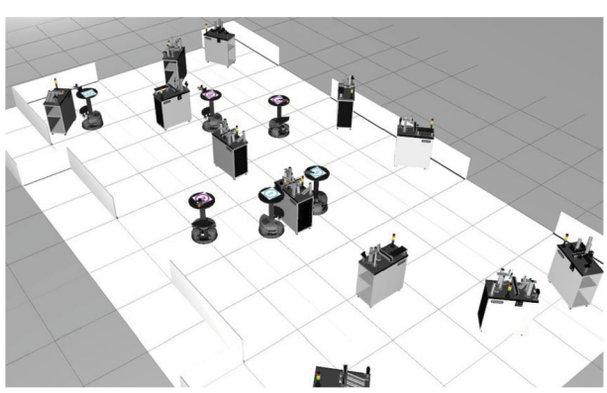


Fig. 1. Rendering of the RCLL setting

Kohout, Peter, Institute for Software Technology, Graz University of Technology, Inffeldgasse 16b, 8010 Graz, Austria; **Bortoli, Marco,** Institute for Software Technology, Graz University of Technology, Inffeldgasse 16b, 8010 Graz, Austria; **Ludwiger, Jakob,** KNAPP Industry Solutions GmbH, Gewerbeparkstraße 17, 8143 Dobl, Austria; **Ulz, Thomas,** KNAPP Industry Solutions GmbH, Gewerbeparkstraße 17, 8143 Dobl, Austria; **Steinbauer, Gerald,** Institute for Software Technology, Graz University of Technology, Inffeldgasse 16b, 8010 Graz, Austria (E-mail: steinbauer@ist.tugraz.at)

termediate products to machines refining them. Usually teams use some central server that collects information from the MPS, the machines as well the robots and coordinates the tasks. The products are mimicked by a stack of bases, rings, and caps. The configuration is flexible and determines the complexity of a product. In general several refining steps of intermediate products by different machines are needed to obtain a final product. A central agent randomly generates product orders with varying configurations and delivery windows. This orders are communicated to the teams' server that need to derive a production schedule and to distribute the tasks among the robots and machines. Based on the complexity of the product and if the delivery window was met points are awarded to the teams. For the most complex products up to 10 different steps like fetching and delivering material to machines are needed. Some of them might be parallelized or rescheduled in order to optimize the awarded points. The goal is that teams develop a production strategy that maximizes the points awarded in a 17 minute production phase.

3. Challenges

The interesting aspect of the setting of the RCLL is that it resembles a flexible on-demand production site which can be easily replicated while it abstracts away some aspects of a full production site where physical changes are made to the product (e.g. milling). By providing this setting the RCLL posts challenges in the full range from Robotics over communication and multi-agent systems to planning and scheduling. We like to list some of the challenges here:

1. Robotics:
   - safe autonomous navigation
   - precise manipulation of small objects
2. Multi-Agent Systems:
   - reliable communication between agents
   - reliable execution of tasks
   - distribution of sub-tasks
3. Planning and Scheduling:
   - obtaining a general production plan
   - optimization of production schedules
   - maintaining resources - material or machines
   - tracking of pre-assembled products
   - reaction to issues during the execution

4. Benchmarking system

The basic idea of the competition is that benchmarking is based on playing a number of different games and to compare the collected points. Each game is unique in the sense of machine layouts and configuration of the requested products. To ensure comparability between different games, the difficult level is kept equal by asking for orders with same complexities, which are determined by the number of production steps and the need for additional materials. The collected points depend on the complexity of the product and if the delivery window was met. The benchmarking and the execution of a game is automated to a high degree, and the execution and the scoring of a game is done mostly automated. Teams may insert and remove robots on demand. But once inserted into the game the robots need to be autonomous. In order to also evaluate the robustness of the production system randomly machines are going out of service to trigger a re-planning of the schedule. Using a physical setting poses a lot of challenges related to hardware and interaction with the real world. Thus the evaluation results also depend on how well a production system deals with these aspects. In order to allow also focusing on the planning and scheduling aspect the RCLL is available as a pure virtual simulation where the basic interactions and the navigation are atomic actions. The competition is part of the ICAPS planning competitions and allows researcher from the planning community to test their planning and plan execution in a challenging production setting [6].

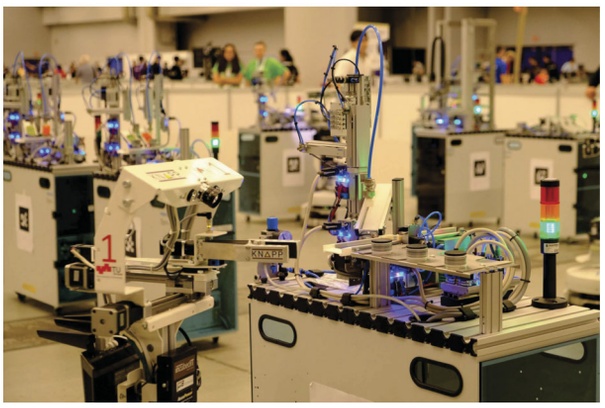


Fig. 2. Robot interacting with a machine

5. The GRIPS use case

In this section we present the approach developed by the team GRIPS (Graz Robust Intelligent Productions System) to face the RCLL challenge. In Sect. 5.1 the used robot system is briefly described, while in Sect. 5.2 we are looking into the software architecture.

5.1 Robot platform

The three mobile robots used by each team are based on the Robotino 3 platform by Festo Didactis, a robot system indented for teaching purposes. Figure 2 depicts the upper part of robots of GRIPS team. The robot is equipped with a standard 2D lidar used for navigation and a specialized 3 axis gripper for manipulating the products. 2 monocular cameras are used for identifying and localizing machines (front camera) and products (top camera). A dedicated PC running the on-board software completes the setup.

5.2 Software architecture

The GRIPS architecture is based on the main aspects required by a multi-robot system, namely (1) planning and scheduling, (2) plan refinement and execution, (3) behavior and control, and (4) low-level functionality. In the following sections we are going to show selected topics for all aspects, except low-level functionalities such as navigation. A general overview of the software architecture used by GRIPS can be seen in Fig. 3. It resembles a classical three-tier architecture [2] with a deliberative layer (high-level), an executive layer (mid-level) and a behavioral layer (low-level).

The planning and scheduling instance is deployed on a so-called teamserver which has global knowledge of the current game, comprising information from the RCLL referee box such as requested orders and from all active robots such as the status of task execution. The teamserver controls all robots and interacts as a gateway between them and the referee box. The teamserver forms the high-level control and runs on an external computer. The modules running on the robots comprise an executive (mid-level) as well as a behavior and control module and and low-level functionalities (low-level).

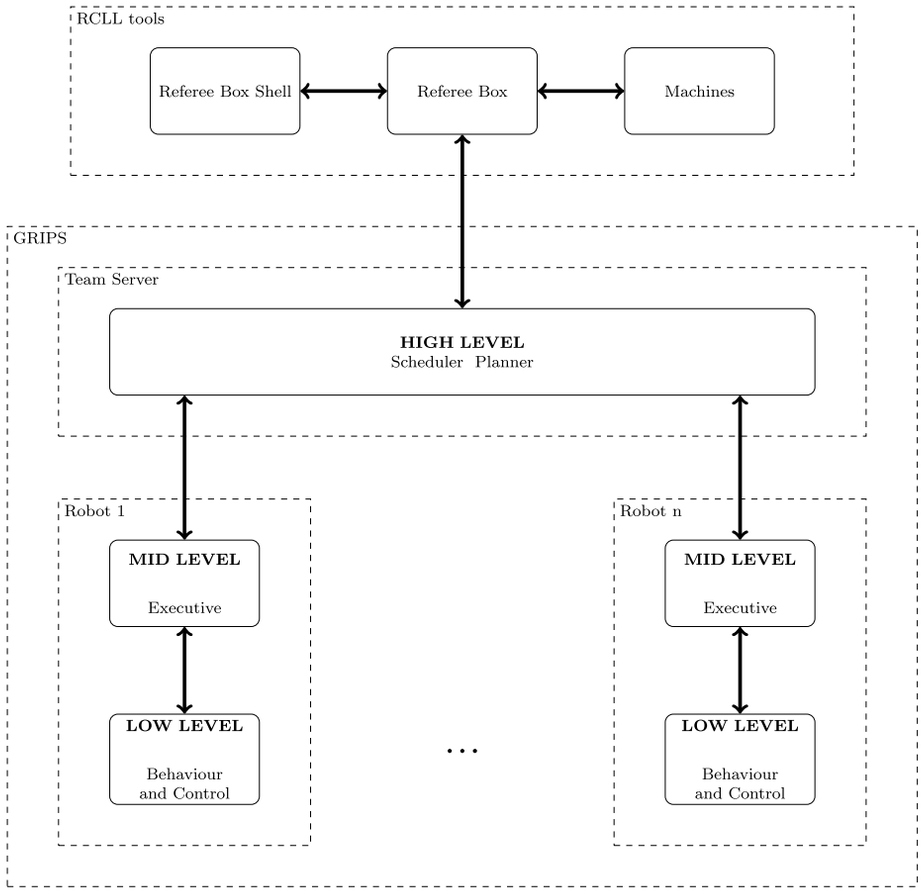


Fig. 3. Architecture of GRIPS software

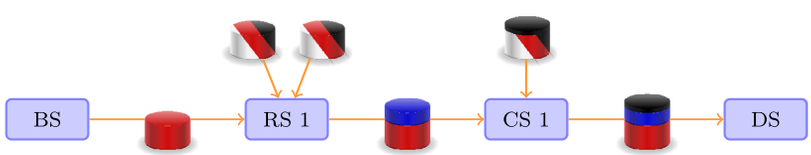


Fig. 4. Example production chain for an order of complexity *C1*, adapted from RCLL rulebook [1]. This order requires 2 additional workpieces (consumables) at the ringstation (RS1) and a cap loaded at the capstation (CS1)

The teamserver is implemented using Java and the Spring framework. The executive layer is realized using the OpenPRS BDI architecture [3]. The low-level functions are implemented in C++ using the Robot Operation System (ROS). The entire software (simulation and robot) is organized in docker containers for easy deployment.

5.2.1 Planning and scheduling

Planning and scheduling in our approach is based on splitting any order that is received from the RCLL referee box into sub-tasks that cannot be split further. This idea is inspired by the concept of hierarchical task network planning [4]. In the system, we distinguish between two sub-tasks categories, since every robot is only able to carry a single item: *GET* and *DELIVER*. A *GET* task consists into going to a machine and retrieving an object from it, after some preparation signal was sent to the machine. During a *DELIVER* task the robot carrying an object navigates to the proper machine in order to accomplish the final delivery or to add some piece (ring/cap) to the carried partial product.

Since getting and delivering a workpiece reasonably needs to be done by the same robot, this specific choice of sub-task types might seem counterintuitive. However, by separating the pickup and deliver process, machines can be freed from a workpiece that would otherwise block the machine for other robots.

Task generator Any order received from the RCLL referee box specifies the color required for each workpiece used in the production process. Therefore, any order implicitly defines which machines need to be used. An example production chain for an order of easy-to-medium complexity C1 is shown in Fig. 4. In this configuration, one ring needs to be mounted before the cap. Based on this production chain, our scheduler creates the required sub-tasks and the corresponding dependency graph using the ideas of HTN refinement. The resulting dependency graph for the production chain shown in Fig. 4 is then depicted in Fig. 5. As can be seen there, sub-tasks belong to one of the following three categories:

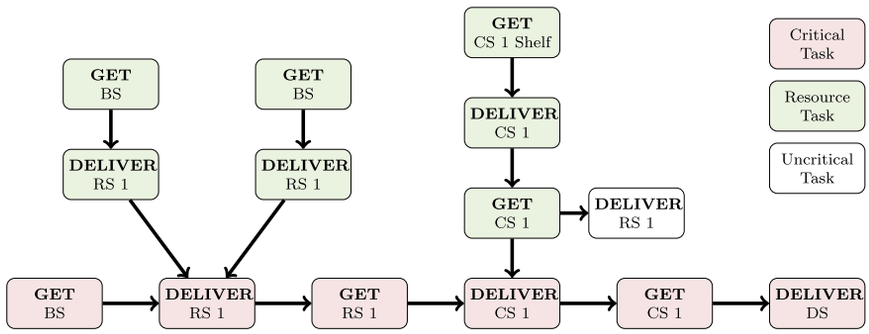


Fig. 5. Dependency graph for tasks required to build and deliver a product for the *C1* order that is shown in Fig. 4. Arrows represent ordering constraints for the tasks as well as enabling preconditions. The Uncritical Deliver task is an optimization bringing the unused base from CS1 to RS1, which may require further bases in the future for a different order

– *CRITICAL TASKS* are tasks whose failure compromises the entire product, which is therefore canceled. It may be restarted depending on the current game's context;
– *RESOURCE TASKS* load the machines with workpieces that are required for the assembly of products. If resource tasks fail, the actual assembly of the product is not harmed, but it could be delayed.
– *UNCRITICAL TASKS* neither influence the successful completion of the currently assembled product, nor do they (directly) influence assembly time of that product. However, if successfully completed, these tasks might have a positive effect by speeding up future assembly processes.

Task scheduling The assignment of tasks to robots is based on a request-response approach. Robots that currently do not own tasks request new tasks from the central planning and scheduling instance. A task is assigned to the robot with the following priorities:

1. task of type DELIVER (from a product in assembly) for which a successfully finished predecessor task was already assigned to the same robot;
2. task of type GET (from a product in assembly);
3. starting of a new product if feasible, by checking if a parallel production chain can be found where no machine, besides Base Station (BS) and Delivery Stations (DS), overlap;
4. starting of a dummy task, where the robot keep randomly moving in order to avoid blocking the machines since, according to the rules, robots are not allowed to stay on a single spot for more than a specified time window.

5.2.2 Executive

The bridging between the abstract planning and scheduling and the practical behavior layer is established by an executive layer that runs separately on each robot. The two main functions of the executive are the refinement of the abstract tasks to executable behaviors and the supervision of the entire task execution. The separation of the two functions contributes to the robustness of the overall architecture as the former allows the system to use a flexible abstracted planning approach while the latter allows to react to uncertainties and unexpected situations in the interaction between the physical robot and its environment. We realized the executive layer following the well-known concept of belief-desire-intention (BDI) and procedural reasoning. In order to allow robust and reactive control of robots the approach follows the idea of practical reasoning where the tasks to be fulfilled are represented by goals and goals are pursuit using scripted recipes called procedure. Procedures are represented as directed graph with further sub-goals on the edges.

The interaction with the other parts of the architecture works as follows. Any time the robot becomes idle it requests a new task from the planning and scheduling component. The tasks assigned to a robot by this component are mapped to configurable goals. Currently we have corresponding goals for the get, delivery, and dummy task with corresponding hand-crafted procedures. The executable basic behaviors like navigating to a given position, alignment at a machine, or grasping an item are represented by primitive goals that lead directly to a behavior execution. The physical execution is realized using the action server concept of the Robot Operating System (ROS) [7]. Each primitive goal is wrapped by a safe version of the original goal to achieve dependable execution. These goals comprise additional hand-crafted monitoring and fault recovery recipes. These recipes are based on an analysis if effects of an action are observable or reversible. These safe goals are reused when structuring the recipes for the top-level goals.

5.2.3 Behavior and control

Several software components are implemented in the behavior and control layer of the software architecture. These components comprise navigation, alignment to machines, identification and localization of machines, and identifying and manipulation of products. In this section, the control strategy which enables the precise alignment of the robot in front of the machine is briefly explained. In order to grasp or place products during production, the robot needs the ability to align itself at very short distances and with very high precision in front of machines, criteria which are not possible to achieve with the usual navigation approaches [5] already implemented in ROS. However, this is a typical task for classical feedback control. The two parts necessary for feedback control are the error computation and the controller design.

To perform closed loop control, the current positioning error has to be computed. The robots are equipped with a laser scanner at the front, which can be used to compute the position of the machine relative to the robot with a very basic clustering algorithm. Based on the estimate of the machine pose, the three errors for position e_x and e_y as well as the angular error e_φ are computed and fed into the controller. At this point, the control algorithm is developed by designing a sliding mode controller for each of the three components of the error (position e_x , e_y and angular e_φ). Sliding mode control was chosen because it is very simple to implement, easy to tune but

also represents a robust control strategy. For further details about the sliding mode controller, the reader can look into [9, 12, 13].

6. Conclusion

Following the common interest of the institutes involved in the GRIPS team in a holistic approach to develop methods for dependable intelligent systems we developed a software architecture that allows flexible and robust execution of the demanded production tasks. The basic idea is to separate different concerns such as abstract planning and scheduling, refinement of task execution, and behavioral control and equip each layer with proper motioning and fault-recovery capabilities. The flexible planning and task assignment paired with robust task execution allowed us to realize more complex products reliably and constantly than in the past competitions. In future work we will aim for the use of a long-term temporal planner, able to better optimize the task scheduling process with respect to the actual request-response approach. Moreover, we like to better team up monitoring and recovery between different layers. Often one layer lacks of sufficient knowledge about the actual situation to make a consistent final conclusion about errors and recovery. Sharing and combining information of different layers may help to address this issue.

Acknowledgement

This work is supported by the Lead Project *Dependable Internet of Things* of Graz University of Technology. Team GRIPS is grateful to its sponsors Knapp AG, incubedIT GesmbH, PIA Automation Austria GmbH, AccuPower GmbH, and myRobotcenter GmbH.

Funding Note Open access funding provided by Graz University of Technology.

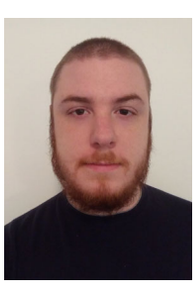
References

1. Coelen, V., Deppe, C., Hoffmann, T., Karras, U., Niemueller, T., Rohr, A. Rules and Regulations - RoboCup Logistics League. http://www.robocup-logistics.org/rules.
2. Gat, E. (1998): On three-layer architectures. In R. P. Bonnasso, D. Kortenkamp, R. Murphy (Eds.), Artificial intelligence and mobile robots. Menlo Park: AAAI Press.
3. Georgeff, M., Pell, B., Pollack, M., Tambe, M., Wooldridge, M. (1999): The belief-desire-intention model of agency. In J. P. Müller, A. S. Rao, M. P. Singh (Eds.), Intelligent agents V: agents theories, architectures, and languages (pp. 1–10). Berlin: Springer.
4. Ghallab, M., Lau, D., Traverso, P. (2004): Automated planning - theory and practice. San Mateo: Morgan Kaufmann.
5. Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., Konolige, K. (2010): The office marathon: robust navigation in an indoor office environment. In International conference on robotics and automation.
6. Niemueller, T., Karpas, E., Vaquero, T., Timmons, E. (2016): Planning competition for logistics robots in simulation. In WS on planning and robotics (PlanRob) at Int. Conf. on Automated Planning and Scheduling (ICAPS), London, UK.
7. Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A. Y. (2009): ROS: an open-source Robot Operating System. In ICRA workshop on open source software.
8. RoboCup Logistics League - Website. http://www.robocup-logistics.org/.
9. Shtessel, Y., Edwards, C., Fridman, L., Levant, A. (2013): Sliding mode control and observation. New York: Springer.
10. Steinbauer, G., Ferrein, A. (2016): 20 years of RoboCup. Künstl. Intell., 30(3–4), 221–224.
11. Ulz, T., Ludwiger, J., Steinbauer, G. (2019): A robust and flexible system architecture for facing the RoboCup Logistics League challenge. In D. Holz, K. Genter, M. Saad, O. von Stryk (Eds.), RoboCup 2018: robot world cup XXII (pp. 488–499). Cham: Springer.
12. Utkin, V. I. (1992): Sliding modes in control and optimization. Berlin: Springer.
13. Utkin, V., Guldner, J., Shi, J. (2009): Sliding mode control in electro-mechanical systems. Boca Raton: CRC Press.

Authors

Peter Kohout
Peter Kohout is bachelor student in Information and Computer Engineering at Graz University of Technology.

Jakob Ludwiger
Jakob Ludwiger receives a PhD in Control form Graz University of Technology and is now with Kanpp Industry Solutions GmbH.



Marco De Bortoli
Marco De Bortoli is a Phd student in Computer Science at the Engineering at Graz University of Technology.

Thomas Ulz
Thomas Ulz receives a PhD in Computer Science form Graz University of Technology and is now with Kanpp Industry Solutions GmbH.

Gerald Steinbauer
Gerald Steinbauer is Asscociate Professor at the Institute for Software Technology at Graz University of Technology.