Specification and Verification of Collaborative Transport Robots

(Invited Paper)

Bernd-Holger Schlingloff
Fraunhofer FOKUS and Humboldt University, Berlin
holger.schlingloff at fokus.fraunhofer.de

***Abstract*—A collaborative embedded system is an intelligent agent in a cyber-physical system which cooperates with others by negotiation to fulfill individual and common goals. Examples are self-driving cars, soccer-playing robots, or adaptive production plants. In this contribution, we present the industrial case study of autonomous transport robots in factory environments. In our setting, the robots collaborate by competing for transport jobs issued by the production machines. Each robot calculates its individual cost incurring with the job (in terms of distance, time, energy, wear and tear, etc.) and places a bid based on this cost. Then a distributed voting takes place, where the lowest cost bid wins the job. Here, we present our results of specifying and verifying this scenario. We collected requirements via user stories for the scenario, formulated these in suitable specification languages, designed executable models as simulation environments, and used statistical model checking and runtime monitoring for analysing the scenario. We argue that for different aspects of the case study, different analysis methods are to be used. However, all of these methods can make use of the fact that the goals of the individual agents coincide. Our results indicate that by the individual optimization of the cost function, reliability and performance of the collaborative group increases. We believe that this result is typical for a large number of similar systems.**

***Index Terms*—embedded systems; collaboration; specification; verification; transport robots; AGV; model-based design.**

I. Introduction

Embedded systems are ubiquitous in our society. It is estimated that in 2017, around 75 to 100 billion embedded systems were in use world-wide [1]. A growing trend is that these devices are becoming more and more interconnected, communicating via wireless links. This is the enabling factor of emerging paradigms such as the Internet of Things, Industry 4.0, Smart Cities, Connected Healthcare etc. Common to all these paradigms is that a group of embedded systems cooperates to achieve certain common goals. We call a cyber-physical system with this architecture a *collaborative system group* (CSG), and the individual agent in this group a *collaborative embedded system* (CES). In this paper, we argue that the goal-directedness of each CES can be used to facilitate validation and verification of the CSG. In contrast to general multi-agent systems and multi-player games, in a collaborative system the goals and intentions of individual components coincide. Therefore, a global strategy for the CSG can be comprised of local strategies for the individual CES. We believe that this holds for a large group of industrial systems where the overall goal is reached by collaboration of different actors.

Fig. 1. Transport robots in a factory

This article is structured as follows. First, we describe our case study of autonomous transport robots. Then, in section III we specify goals and targets for the CES in the case study. In section IV, we model different strategies for achieving the objectives, and report on our verification activities for these strategies. Finally, we summarize our experiences and point to some future work.

II. Autonomous Transport Robots

Our case study deals with a fleet of autonomous transport robots in a factory environment [2]. These driverless vehicles are used for loading and unloading production machines, to ensure a timely delivery and disposal of material to different sites in a production process. In contrast to traditional automatic guided vehicles, transport robots not only move without a human driver, but can navigate freely in their environment, without being bound to a fixed track. A typical fleet consists of 4–20 robots, each with a carrying capacity of 50–200 kg with a speed of 0.5–2 m/s [3]. Figure 1 depicts these robots in action, carrying buckets of liquid. In the factory, there are a number of production machines which need raw materials as input, and produce assembled goods as output. The output of one machine may be needed as input for another machine. There is a storage where the raw materials and assembled products are deposited. The task of the transport robots is to service the production machines, by loading and unloading the necessary materials. Each machine signals when it needs being served. The challenge is to distribute the transport jobs

between the robots such that all jobs are accomplished in a satisfactory way. That is, each machine should be served with a minimal waiting time, the load distribution should be balanced, the battery usage should be within defined boundaries, etc.

Here, we assume that each robot is an autonomous agent which can decide on the jobs to accept, the route to drive, the time to recharge, and various other things. There is no global controller directing the whole fleet. The robots can communicate via a wireless network, but this communication is unreliable. There may be spaces where the robot is unreachable or can be reached only by some other robots.

Furthermore, the transport system as a whole must be robust and flexible: It may be the case that roads or docking points are blocked, robots fail or leave the fleet, and additional robots are joining the collaboration. In our current setting, all robots have the same capabilities; however, the design and analysis methods should be able to handle heterogeneous fleets as well.

The robots perceive their environment with the help of a laser scanner. Besides providing functional safety by interrupting the power flow to the motors if anything comes within a certain distance, the laser scanner delivers an accurate 2D-image of the vicinity of the robot. The robot can use this image for localisation. It has a pre-recorded map of the factory, and matches the image with this map to find the maximal likely position.

Using its own position and the position of a goal destination, the robot can calculate an optimal path towards the goal. The map charts the location of docking and charging stations, parking spots, navigable spaces, and one-way roads. An example of a map with navigable area, loading points and data scanned during operation is given in Figure 2.

If the planned path of a robot is blocked, e.g., by a human standing in its way or another robot crossing the route, the robot has to recalculate its path and circumvent the obstacle.

III. Specification of Collaboration

For specifying the different aspects of this scenario, we employed several semi-formal and formal methods. First, we collected *user stories* [4], which describe the whole transportation system from the viewpoint of human stakeholders.

A user story describes for users in a certain role an aim which they can achieve by applying the system at a certain time, as well as the reasons for pursuing this aim. User stories may be exemplified via use case descriptions, formalized in UML use case diagrams and sequence charts.

For example, the user story "Decentralized order management" describes how the robots determine which one of them accepts an order. It can be formulated as follows.

> As a *transport system operator* (role) I want the system to *decide autonomously which robot accepts a transport job* (what) whenever *a job is issued by a machine* (when), such that *there is no need of a central control* (why).

This user story is exemplified by the following steps [5].

	Who?	What?	When?	Why?
1	Machine	Broadcasts transportation need to robots	Every time a machine has support or dispose need (may be in advance and/or may be with priority)	The production process of the machine is not allowed to stop
2	Every Robot	Calculates a bid for this transport (may be based on individual cost and/or other criteria)	When a new transport need is notified	To get the information which robot fits the best for this transport
3	Every Robot	Determine winner by distributed leader election algorithm	After bidding	
4	Robot	Bid winner adds the transport to its own transport queue	When won a bid	That the transport need is satisfied

From these user stories, we derived objectives for the system. An *objective* is a specific requirement describing a purpose of the system. In particular, the main objective describes why the system is being built. Objectives can be structured into a hierarchy, where lower levels support higher levels. Moreover, they can be ordered according to their importance, or level of contribution to the topmost objective.

Objectives can be categorized as *goals* and *targets*. A goal is an objective with a clear criterion whether it has been reached or not, whereas a target is an objective which can also be partially met, to a higher or lower degree. A goal is a certain state of affairs which an agent strives to reach or maintain. An agent can be close to a target, but being close to a goal is the same as missing it. Therefore, goals usually have a long-term character, whereas targets are frequently re-evaluated.

For cyber-physical systems consisting of several independent agents, we have to distinguish between objectives (goals and targets) for the collaborative system group (CSG) and for the individual collaborative embedded system (CES). The individual objectives should support the group objectives.

In our case study, the overall objective of the CSG is to provide transportation services to machines. The most important high-level goal is to keep the maximal waiting time of each machine below a given threshold. That is, if the machine emits a request for a transport job, then it will be serviced by exactly one robot within this threshold. The rationale for this goal is that production machines often have a buffer for incoming and outgoing materials. If the input buffer is empty or the output buffer is full, the machine will stop its operation. This needs to be avoided. A related high-level target is to minimize the average waiting time of machines, in order to cope with varying production speed. Low-level targets include

- *robustness and fault tolerance*, e.g., being able to deal with failures of (unloaded) robots, being able to circumvent temporary road blocks;
- *scalability and flexibility*, e.g., being able to dynamically integrate new robots into the fleet, and being able to adapt to changes in the factory topology;
- *efficiency and durability*, e.g., balancing the usage of robots for equal wear and tear; and
- *security*, e.g., intruders and traitors cannot bring down the system.

These fleet-related goals and targets must be complemented

Fig. 2. Factory map with scanned floor plan, nogo-areas (grey), and docking and charging stations

with individual objectives for each CES. The topmost goal for each robot is to accomplish each transport job it has accepted, if this is within its capabilities. A related target is to accomplish the accepted transport jobs as fast as possible.

In order to support the topmost global goal "each request will be serviced", corresponding objectives for the individual robots must be set. For example, an individual target could be to service as many requests as possible.

However, in isolation, this target might be too coarse, as it might lead robots to "self-destructive" behaviour such as neglected charging, extensive wear and tear, congestion of roads in the factory, etc. As an example, consider the case when a robot has a low battery level which would allow to finish one more transport job, but it would risk running out of energy on the subsequent way to a charging station. Here, we have a conflict between different individual goals and targets. Should the robot prioritize the target of servicing as many requests as possible over the goal of never running out of battery? This shows that the targets must be refined with an appropriate strategy which takes all objectives into respect.

Further goals and targets include

- keeping within designated floor areas,
- being able to cope with obstacles and road blocks,
- keeping battery level at 40–70%,
- minimizing the occupation time of docking and charging points,
- minimizing the number and length of empty trips, and
- avoiding rests outside designated parking areas.

For the formal specification of goals, temporal logics can be used. A classical example is the property "for every request there is a subsequent response". This is written in linear temporal logic (LTL, [6]) as follows.

$$\Box(request \rightarrow \Diamond response)$$

In our setting, properties refer to real-time values. Therefore, timed temporal logics are necessary. The property that every request for service by a machine is fulfilled within 60 time units by some robot can be written in metric temporal logic (MTL, [7]) as follows.

$$\Box(request(m_i) \rightarrow \Diamond_{<60} \exists r_k \; at(m_i, r_k))$$

Here, the existential quantifier is a finite disjunction ranging over the finite set of robots. Other goals need spacial, epistemic, or strategic operators for formalization. It is much harder to express quantitative targets in classical or modal logics. If the bounds are made explicit (as in the example formula above), we can use these bounds in formulas. For example, we can specify performance in Weighted Metric Temporal Logic (WMTL, [8]). This logic contains an operator $\mathbf{P}$ which returns the probability of a statement within a certain time period. As an example, let the response time be the time difference between the time when a job is created and the time when the job is finished. The property "The response time within the first 1000 time units shall be less than 450 time units in 80% of all requests" can be written in WMTL as follows.

$$(\mathbf{P}_{<=1000}(\Box(Job.Active \rightarrow (Job.clock \leq 450))) \geq 0.8)$$

However, we are not completely satisfied with these specification logics. In a way, they do not allow to adequately translate the natural-language formulation of the targets. The numerical borders (1000 time units, 450 time units, 80%) are introduced artificially for the purpose of specification, they do not appear in the original target. Therefore, we are currently looking for other means of formalization which allow to precisely formulate also fuzzy requirements in an uncertain and dynamic context.

IV. Verification and Validation

In the most general meaning, a *strategy* is an operationalization of goals and targets. Given a state of an agent, a strategy allows the agent to make a decision as to which action to perform. Following an adequate strategy should eventually lead to attaining the objectives. Strategies can be formulated in state-transition-based modelling formalisms. There have been many suggestions of such formalisms, from Kripke structures and finite automata to UML state machine diagrams.

We derived various models for our case study, reflecting the defined objectives. For example, a strategy supporting the topmost goal of servicing each machine in time consists of an auctioning mechanism for issued transport requests. This strategy can be formulated in natural language as follows.

Each robot maintains a local queue of accepted transport jobs and estimated completion times. If a machine issues a

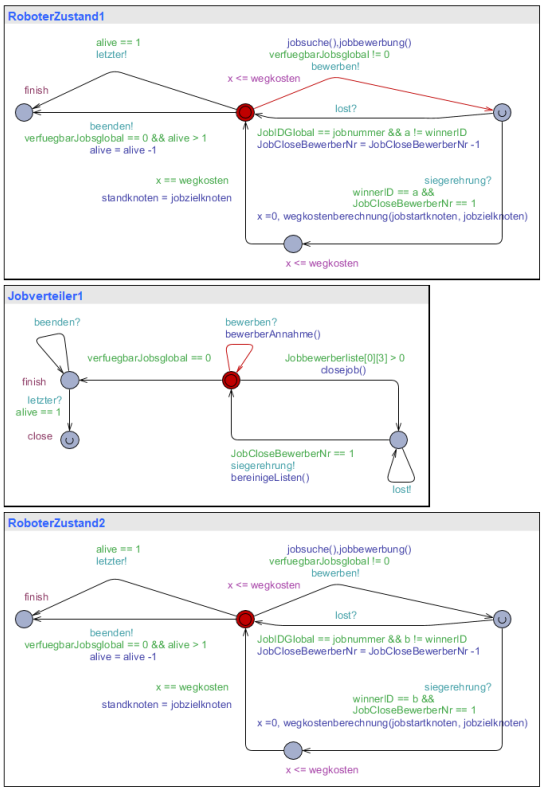

Fig. 3. Model of transport system with two robots

new request, the robot calculates an estimated arrival time at this machine. The job mileage is the distance between the last position in this list and the location of the machine. The estimated arrival time is the estimated completion time of the last job in the task queue and the estimated travel time for the job mileage. If this estimated arrival time is within the deadline of the job, the robot places the job mileage as a bid. Then, the robot waits and collects other bids. After a certain deadline, the robot selects and communicates the lowest bid. If no bids or more than one lowest bid arrived, this is an error an bidding must start again. Otherwise, if the robot has placed the lowest bid itself, the job is appended to the task queue.

This strategy does not take into respect power consumption, battery charging, wear and tear, and other targets. We formulated and modelled further strategies, e.g., the following.

- Random: In the bidding, each robot bids a randomly chosen amount.
- Shortest time: Each robot bids the estimated arrival time at the machine.
- Highest energy: Each robots bid its current battery level (highest bid wins). A variant is to bid the estimated energy after completing the last job in the task list.
- The bidding sum is a function of the job mileage, and the total mileage of the robot.
- First-come-first-served: Each robot records a queue of all issued requests. Whenever a robot becomes idle, it takes the first unserved request from this list and signals it to the others.

These strategies cover the assignment of tasks to robots. Of course, the other objectives must be supported by appropriate strategies as well. For example, in order to realize the global goal of fast and reliable service, we have to model a mechanism with which the robots communicate information amongst themselves. Each agent should distribute information about its environment, e.g., obstacles, road blocks and map updates. Moreover, it should also convey some information about itself, e.g., current position and planned route.

For the formalization of these strategies, we used transition systems, timed automata, stochastic timed automata, and interpreted systems programming language. In general, a model of the whole systems consists of three parts:

- A model of system, i.e., the robots and their behaviour,
- a model of the static environment, i.e., the floor plan of the factory, and
- a model of the dynamic use of the system, i.e., a description of the transport jobs.

As an example, Figure 3 shows (part of) an UPPAAL model with two robots and one transport job generator, where the robots follow the auctioning mechanism described above [9]. The floor plan of the factory environment is modelled as a weighted symmetric finite graph, where the distance between machines is calculated by Dijkstra's shortest path algorithm.

Given the formalization of a goal in computation tree logic (CTL), this model allows its automatic verification by model checking. Since the modelling formalisms we used do not allow the dynamic creation of agents, we had to fix the number of robots to a fixed constant. For models of decent size, the verification is quite fast. As a typical example, with two or three robots and five to ten machines, most properties can be verified within a few seconds However, there is a state space explosion with respect to the number of robots and docking stations. Since there is little independence in the parallelism, partial order reductions did not really help to reduce the complexity. Nevertheless, since in practice both of these parameters are limited, this is a viable approach.

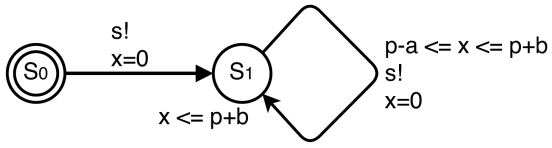

Fig. 4. Timed automaton for production with jitter

To a certain degree, stochastic models can help to optimize certain parameters of a system. We modelled a factory with a narrow passway between the storage and the production machines, which can only be occupied by one robot at a time [10]. All transport jobs have to use this passway. To avoid collisions, a robot has to wait while another robot is passing through the passway. If several robots want to pass, they have to coordinate who is allowed to pass first.

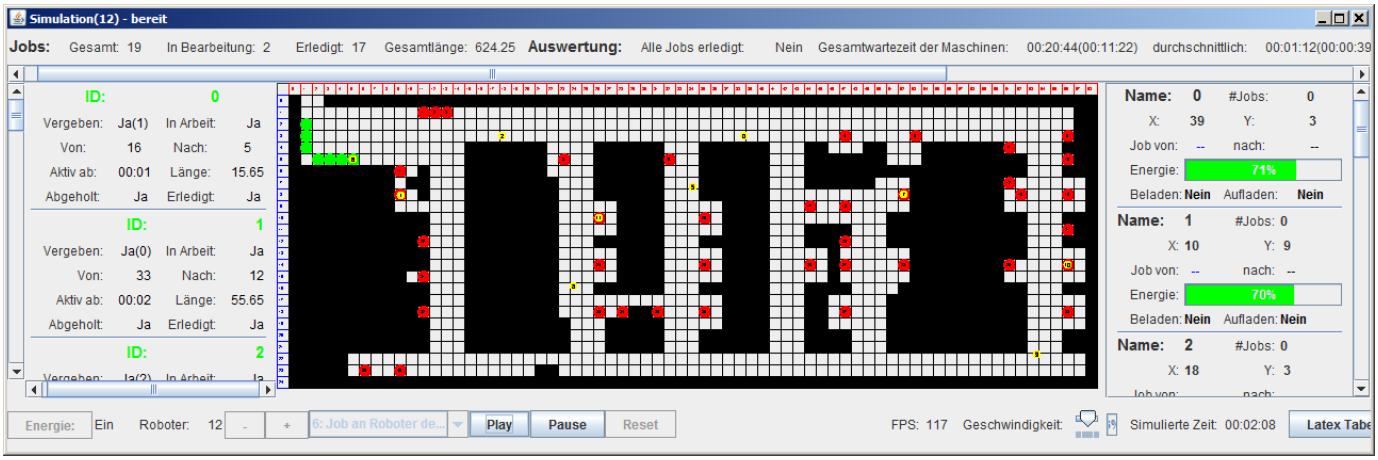

Fig. 5. Simulator for various strategies

The dynamic usage model determines that all machines cyclically issue requests, each with a certain frequency x_m. We attached a jitter $[a, b]$ to the frequency x_m. That is, if a request is issued at time t, the next request will be issued at any time in the interval $[t+x_m-a, t+x_m+b]$. Figure 4 shows a timed automaton for this behaviour. Model checking of appropriate WMTL formulas revealed that the model is highly sensitive to this jitter. Whereas for certain scenarios where $a = b = 0$ congestion could be avoided, with $a > 0$ and $b > 0$ the worst-case performance was drastically reduced. Model checking, in contrast to simulation, considers all possible executions, and thus can reveal best-case and worst-case behaviour.

However, also for stochastic timed automata, a state space explosion occurs. In order to be able to evaluate the behaviour of the CSG with many robots and stations under different strategies, we implemented a tailored simulator [11]. A screenshot of this simulator is shown in Figure 5. Here, a transport job is given as a tuple $\langle$`time, from, to`$\rangle$. The list of transport jobs is depicted on the left side of the simulator window. The robots, their current occupation status and battery level are given on the right side. The main area of the simulation window shows the floor plan, docking and loading stations, and a visualization of the robot's movements. The screenshot shows a map corresponding to the floor plan in Figure 2. Simulation mode, number of robots, strategy and simulation speed can be set with appropriate buttons. The simulator logs the overall time until all jobs are completed, the completion time for each jobs, idling time of machines, average load and travelled distances per robot and in total.

Extensive simulation runs yielded two main results. Firstly, the ranking of different strategies depended strongly on which target metrics was used: Shortest overall time, shortest maximal idling time, best throughput, etc. Secondly, for each strategy there were scenarios where they performed particularly good, and others where they failed. However, some strategies (notably, "random") scored almost always bad, whereas others (notably, "job mileage" and "shortest time") were mostly in the top group [11].

From these results, we draw the following conclusions. Finding the "right" scheduling strategy is a high-dimensional optimization problem which has many Pareto-optimal solutions. That is, it is not possible to improve in one result parameter without loosing in others. Moreover, since the simulation depended on many parameters, maybe a "learning" strategy which adapts the robot's behaviour to different contexts might outperform all fixed strategies. Formal verification of learning systems, however, is still in its infancy.

An alternative to formal verification of static models and programs is dynamic verification at runtime. Online monitoring is a technique where the observed traces of a running system are compared to a formal specification. Traces can be obtained from a prototypical implementation, or from a simulation as described above. A challenge is to design a monitoring system which can detect and flag problems early, ideally even before they occur. That is, the monitor should raise an alarm even while the system is acting normally, if it has a tendency to drift into the exceptional behaviour.

To this end, we extended the semantics of metric temporal logic MTL (see above). Basically, we consider traces of finite length, which are processed one step after the other. A formula in a timed trace at a certain instant not only can be *true* or *false*, but the truth value additionally can be any real number. As long as the truth value of a formula in a model is not determined according to the standard semantics, we assign it a "likelihood" of being satisfied. This "likelihood" is calculated from the distance of the deadlines in the formula to the end of the trace, and the respective values for the sub-formulas.

We implemented an algorithm for monitoring our extended semantics [12]. For the evaluation of this algorithm, we collected traces from (a centralized version of) our transport robot case study. These traces covered a duration of several days up to a week of operation, and contained more than 10^6 timed events. By analysing the response time of the transport system to issued requests, we found a quite high variance of

this value (between 1 and 100 min). Evaluating the MTL response property given above (every job will be fulfilled within 60 sec) revealed that property violation tended to "build up": several "near misses" were followed by a definite miss. This could not have been found by classical boolean-valued monitoring methods.

V. Conclusion and Further Work

We presented results in the specification and verification of collaborative embedded systems. As a case study, we described an indoor logistics system consisting of autonomous transport robots in factories. For the specification of the system, we used a stepwise refinement approach: From purposes via objectives to strategies. We specified the aims of the system with user stories in controlled natural language, augmented by scenarios as sequences of steps. From this, we derived formal objectives for the system. We categorized the objectives as goals or targets, depending on whether they are purely qualitative or also quantitative. Furthermore, we showed how to formalize the goals in various temporal logics. Then, we identified strategies supporting the defined goals and targets. We formalized the strategies in different state-transition-based modelling formalisms, and used simulation and model checking to analyse these models. Finally, we showed how to apply monitoring techniques to analyse long execution traces of the system for the accumulation of problems.

From our results, we can draw the following conclusions. Firstly, there is no "one size fits all" method for the formal analysis of such complex systems. We used different methods, e.g., timed automata, for different verification goals, e.g., correct timing. For the defined targets, we used quantitative analysis methods such as stochastic simulation and probabilistic model checking. However, these methods forced us to introduce artificial bounds. Therefore, we are trying to find method which can give precise results also for fuzzy requirements and approximative targets.

Secondly, our analysis was facilitated by the fact that we were dealing with collaborative rather than competitive systems. In this paradigm, the individual strategy of each agent contributes to the aims of the whole system. The environment has no "strategy" to adverse the outcome. Therefore, by an individual optimization, the performance of the collaborative group increases. We believe that this observation is typical for a large number of similar systems. However, we could not yet make this statement precise. Therefore, we are looking for strategic logics which allow to formally characterize collaboration aspects in embedded systems. This might improve the development process by an automated synthesis of collaborative strategies.

References

[1] https://www.quora.com/How-many-embedded-systems-are-there

[2] H. Schlingloff, H. Stubert, and W. Jamroga: Collaborative embedded systems – a case study. In: Proc. EITEC 2016 - 3rd Int. Workshop on Emerging Ideas and Trends in Engineering of Cyber-Physical Systems. CPS-Week, Wien, Apr. 2016.

[3] http://www.insystems.de/en/produkte/proant-transport-roboter/

[4] M. Cohn: User Stories applied for agile software development. Addison-Wesley, Amsterdam 2004.

[5] J. Zernickel, S. Dannat, A. Schmiljun, H. Stubert, and J. Samuel: CrESt Deliverable UC.AP4.D1, Cooperating transport robots. Internal Report, Aug. 2017

[6] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi: On the temporal analysis of fairness. Proc 7th ACM POPL, 1980.

[7] R. Koymans, Specifying real-time properties with metric temporal logic. Real-time systems, vol. 2, no. 4, 1990.

[8] P. Bulychev, A. David, K. Larsen, A. Legay, G. Li, and D. Bøgsted Poulsen. Rewrite-based statistical model checking of WMTL. 3rd RV, Istanbul, 2012,

[9] F. Walter: Modellierung und Verifikation von Sicherheit und Lebendigkeit eines Systems mobiler Transportroboter mit Modellprüfung zeitbehafteter Automaten. Bachelor's Thesis, Humboldt Universität zu Berlin, Institut für Informatik, 2016.

[10] R. Arai, H. Schlingloff: Model-based performance prediction by statistical model checking, an industrial case study of autonomous transport robots. In: Proc. 25th CS&P 2017 - Concurrency, Specification and Programming. Warsaw, Sept. 2017.

[11] F. Sitzmann: Simulation und Vergleich der Effektivität verschiedener Job-Scheduling-Verfahren für autonome Transportroboter. Bachelor's Thesis, Humboldt Universität zu Berlin, Institut für Informatik, 2018.

[12] F. Lorenz and H. Schlingloff: Online-Monitoring Autonomous Transport Robots with an R-valued Temporal Logic. Submitted to CASE 2018.

Acknowledgment

This work was funded by the BMBF Project CrESt, FKZ 01|S16043 G&E The author would like to thank the industrial partner InSystems Automation GmbH for providing the case study of this work. In particular, Jan Stefan Zernickel from InSystems provided valuable feedback on an early version of this paper.