SOFTWARE-IN-THE-LOOP TEST BENCH FOR THE EVALUATION OF THE COORDINATIVE FLEET CONTROL APPROACH IN INTRALOGISTICS

Florian Rothmeyer and Johannes Fottner
Chair of Materials Handling, Material Flow, Logistics
Technical University of Munich
D-85748 Garching bei München, Germany
E-Mail: florian.rothmeyer@tum.de

KEYWORDS

Discrete-event Simulation, Robotics, Intralogistics, Fleet Management, Guidance Control, Autonomous Mobile Robots (AMRs)

ABSTRACT

A system coordinator for the intelligent forwarding of transport orders enables the combination of several fleets of intralogistic transport vehicles such as robots in one material flow system without them having to exchange information with each other. Evaluating the operation of this coordinator requires a simulation environment that represents all adjacent systems, creating a Software-in-the-Loop (SiL) environment. This includes upstream systems, e.g. enterprise resource planning or manufacturing execution systems, several fleet controllers as well as a material flow system with all vehicles including their interactions and possible failures. In this article, the conceptual design of such an SiL system is presented. Additionally, a proof-of-concept shows the fundamental functionality by comparing the perfomance of two different configurations of the system coordinator. While the results are plausible relative to each other, an improvement of the system configuration is needed for the absolute validity of the values.

Python is used for the system coordinator and the analysis, openTCS for the fleet controllers and Tecnomatix Plant Simulation for the material flow simulation. The communication in between uses HTTP and raw TCP packets, respectively.

I. INTRODUCTION

Intralogistics is currently experiencing a trend toward comprehensive automation. This is reflected in a growth of 45% in the number of transport robots used from 2021 to 2022 [4]. Companies, especially small and medium-sized enterprises (SMEs), are facing high pressure to produce efficiently, which is due, among other things, to intense competition and the increased number of variants in small quantities. In order to be able to produce efficiently and flexibly at the same time, automation is increasingly being used, including in intralogistics. In this context, mobile robots (MR) such as automated guided vehicles (AGVs) or autonomous mobile robots (AMRs) are again the most flexible solution, as they require little infrastructure and can be quickly adapted to changing environments. Another driver for the introduction of robotics in intralogistics is the shortage of skilled workers. Since many transport requirements are traditionally carried out by manually operated industrial trucks, the shortage of personnel can be countered relatively easily by replacing the vehicles with those with automatic or autonomous driving functions.

Transport robotics is therefore an important part of the development of companies that want to remain competitive. In SMEs, however, the introduction of MRs is hampered by limited innovation capability, long investment cycles, high competitive pressure, and the complexity of existing production environments. As a result, many companies can only introduce MRs successively for individual, delimited transport needs.

The steadily increasing variants of MRs favor this, but also lead to an increasing number of different MR types being used in parallel in a material flow system. The extension of existing MR systems by further vehicles with new transport capabilities is difficult today, because the interfaces are not standardized. New systems are already partly supporting interface standardizations such as those defined by the guideline VDA 5050 [8] . However, these do not cover all the necessary communication content and are not prepared for the integration of existing systems.

Consequently, an optimization problem arises between the goals of low cost, low innovation effort, wide choice of suppliers, and high transport efficiency, as visualized in Figure 1.

This optimization problem can be solved by means of

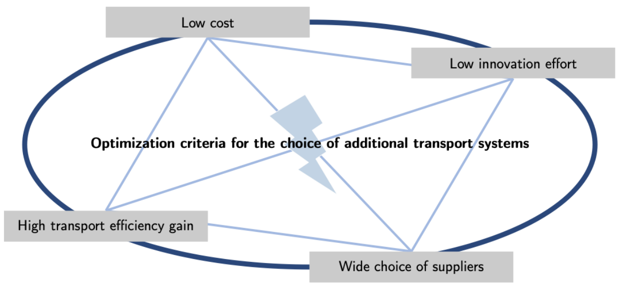
Low cost
Low innovation effort
Optimization criteria for the choice of additional transport systems
High transport efficiency gain
Wide choice of suppliers

Figure 1: Contrary optimization criteria SMEs face when implementing a heterogeneous robot-based transport system.

a novel coordinator: It is placed between ordering systems and master controllers of inventory as well as new systems and takes over an abstraction of the transport systems by forwarding transport orders and optionally performing traffic control. The coordinator as an intermediate link in the chain of systems involved in transport order execution is shown in Figure 2.

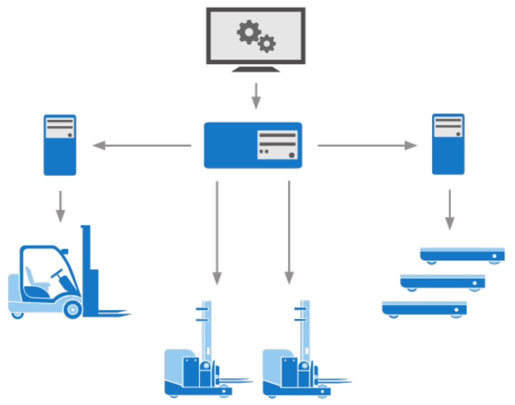

Figure 2: Structure of systems that participate in transport order management and execution in a material flow system with several fleets of transport vehicles following the coordinative approach.

For effective and at the same time economical operation of the overall system with a system coordinator, good dimensioning of the interfaces between coordinator and fleet controllers is essential. Consequently, the algorithm for the selection of the best-suited fleet must be selected wisely to be suitable for the available information.

The concept of the coordinator including its algorithms and interfaces to other systems were developed in the research project EPoSysKo – Development and Potential Analysis of a System Coordinator for the Cross-Type Use of Industrial Trucks [3]. In addition to MRs, also manually operated industrial trucks like forklifts or tugger trains were taken into consideration. On the abstraction level of our research, they behave almost identical to MRs, therefore we will keep using the term "MR" in the following.

The performance of the coordinator in material flow systems with several fleets of MRs can only be determined experimentally due to its complexity. In the research project, this was achieved by simulation. Compared to the implementation in a real system, the integration effort is smaller here. At the same time, the repeatability of experiments under standardized conditions is better. The system coordinator simulation consists of several components that represent the systems involved. Therefore, the architecture around the system coordinator can be regarded as a Software-in-the-Loop test bench.

In this article, these subsystems and the communication among them are presented, as well as the organization and evaluation of simulation experiments. At first, we give an overview of the state-of-the-art regarding simulation of MRs and hierarchical control structures. Subsequently, we explain the conceptual design of the simulation environment. After that, we present the results of a proof-of-concept simulation experiment, that were achieved using a prototype implementation of the simulation environment. Finally, we summarize the results and point out open questions for future research.

II. STATE-OF-THE-ART

At first, we present prior work in the field of robotic fleet simulation. Afterwards, we take a short look at the theory of hierarchical control structures and some algorithms that can be used in robotic fleet management.

A. *Simulation of Fleets of Mobile Robots*

For the evaluation of the performance of intralogistic transport systems, simulation is a widely used tool. These transport systems usually contain several production stations with specific behaviors as well as a high number of vehicles that fulfill the material demands of the stations.

The vehicles can be modeled in a way that the basic physics of the whole process are respected. Berndt et al. used this approach to test a newly developed fleet controller for MRs in intralogistics in a simulation [2]. In their fleet controller software, they implemented special mechanisms for task scheduling, deadlock detection and collision avoidance. The driving commands are sent to robots via hierarchical control loops consisting of one "Mission Control Center" and several "Robot Control Centers". The authors evaluate these mechanisms in a physical simulation using the software Gazebo in combination with the Robot Operating System.

Simulating the physical properties of several robots, each with several complex sensors, in real-time leads to a very high computational effort. To reduce this complexity, intralogistic systems are usually simulated in discrete-event simulations. This type of simulation relies on a more abstract way of modeling processes. A very widespread software tool in the industry for this kind of simulation is called *Tecnomatix Plant Simulation* from Siemens. In comparison to Tecnomatix Plant Simulation, the open-source fleet controller software *openTCS* is very simple, but can also be used for the simulation of transport vehicle fleets. In the following, we show how these two tools have been used previously for research in the field of MRs.

Tecnomatix Plant Simulation. Viharos et al. [10] present a discrete-event simulation model applied to the control of AGVs in an automated assembly system. For this purpose, an exemplary model is first created in Tecnomatix Plant Simulation. The goal is to automate the transport of products within the assembly system to the next station. These products are transported to the respective industrial robots with the help of AGVs. The number of AGVs that can travel through this system is not fixed and can therefore be freely specified. The unprocessed product is loaded onto an AGV at one of two sources and, after passing through the processing steps, is transported via the robots to one of two sinks.

Lienert [5] presents a methodology for simulation-based throughput analysis of AGV-based picking systems. The aim of the methodology is to model different system types of AGV-based picking systems so that a uniform mapping in a simulation environment is possible. Furthermore, a time window-based routeplanning approach is integrated into the methodology. As a simulation environment, Lienert uses the Tecnomatix Plant Simulation software to perform the systematic throughput analysis.

openTCS. Wißing et al. [11] combine openTCS with a self-developed control framework for omnidirectional FTF. Here, the interface for so-called vehicle drivers is used, which allows to implement individual vehicle functions. In this case, the vehicle driver contains the functions required for the exchange of information with other network partners, such as TCP communication. For the smooth processing of transport orders, a so-called Map Manager is located downstream of the openTCS control system, which is informed about obstacles and forwards driving orders in a targeted manner.

Another possible application of openTCS is presented by Ai et al [1]. Here, openTCS is combined with the MQTT (message queue telemetry transport) communication protocol. In this case, the MQTT protocol serves as a means of communication between a station and the transporter. openTCS is used as an instance for the further processing of transport orders. The creation of transport orders is handled by other services such as enterprise resource planning (ERP) systems, manufacturing execution systems (MES) or other systems. The Operations Desk of openTCS serves as a visual overview of the map.

III. CONCEPTUAL DESIGN

As the previous section shows, the authors could not find any references of existing concepts for the simulated testing of a coordinative control system for intralogistic transport systems. In this section, we present a concept that should fill this gap. The whole simulation concept can be broken down into several components and their interactions. We will present a brief summary of the idea behind the system coordinator itself to make the purpose of the simulation environment more understandable. Furthermore, we give an overview of the other involved components as well as their linkages. Afterwards, we take a deeper look at the components *fleet controller*, *material flow simulation* and *ERP system*. Finally, we explain our methodology for evaluating the coordinator's performance by measuring specific parameters.

A. System Coordinator

The goal of the system coordinator is to simplify the combination of several fleets of industrial trucks in a common material flow system. This is achieved by letting the coordinator decide which of the incoming transport orders is forwarded to which one of the fleets of transport vehicles. The coordinator uses the usually incomplete information he gets from the controllers of these fleets for his decision. The more information it gets from the fleets, the better his forwarding decision should get in terms of transport efficiency and avoidance of blocking. It is worth emphasizing that the coordinator does not choose a single vehicle which should fulfill a transport order, but only a fleet.

Hence, the coordinator performs some of the tasks that, according to the Association of German Engineers, are part of a master control system [9]: A disposition is carried out partially, up to the fleet level. Scheduling, i.e. the timing and sequence change of orders, can also be carried out as an option.

The fleet controllers, on the other hand, do not know about the other fleets that are participating in the material flow. They do the scheduling and dispatching of incoming transport orders just as if they were the only system of transport vehicles.

Figure 3 illustrates the sequence of subtasks the coordinator and the fleet controllers fulfill in the coordinative control system for transport tasks. Details about the coordinator concept and the development of its interfaces and algorithms can be found in [3].

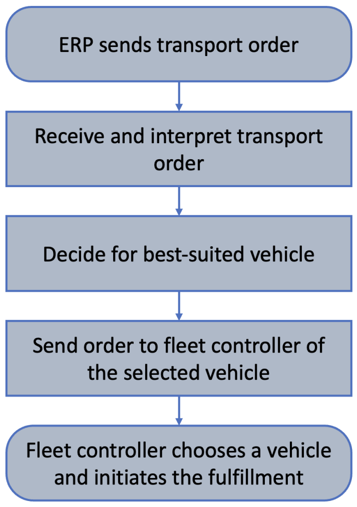
ERP sends transport order
Receive and interpret transport order
Decide for best-suited vehicle
Send order to fleet controller of the selected vehicle
Fleet controller chooses a vehicle and initiates the fulfillment

Figure 3: Sequence of tasks in the coordination process of a transport order.

B. Architecture of the System

For effective and economical operation of the overall system with a system coordinator, good dimensioning of the interfaces between the coordinator and fleet controllers is essential. Consequently, the algorithm for transport order forwarding must be suitable for the type and amount of available information. For analyzing these design parameters, the system coordinator needs to be put in a test bench that allows for reproducible experimental conditions.

We achieve this with a Software-in-the-Loop system. It consists of the components depicted in Figure 4. An **ERP system mockup** sends transport orders to the system coordinator. This is achieved by method calls inside a Python program. The **coordinator** sends each transport order to one of several instances of the fleet control software openTCS via HTTP requests. The **openTCS instances** send driving orders to the simulated vehicles in Tecnomatix Plant Simulation via raw TCP packets. The confirmation of successfully finished orders goes back in the same manner.

C. Fleet Controllers

The software around the system coordinator should work as realistically as possible. Consequently, it is beneficial to use software from the real context for the simulation environment. For the fleet control software, there exists an open source software project called openTCS (open transport control system). It consists of four parts:

- a kernel that computes the decisions regarding scheduling, order-to-vehicle-assignment as well as routes, receives transport orders and sends drive orders to the vehicles,
- an organizational graphical user interface called "KernelControlCenter" for connecting vehicles to the kernel,
- a "ModelEditor" for modeling the network of transport paths and load handling stations that can also import XML-based network descriptions,
- the "OperationsDesk" for observing the operation of the fleet and optionally feeding in transport orders manually.

The kernel also has a web application programming interface for receiving transport orders or returning status information via HTTP requests. [7]

For our Simulation environment, we use several instances of the whole software structure, running on different physical computers which are connected by a local area network (LAN). This is because of possible interferences of several systems running on the same computer. Also, preliminary experiments showed that moving the openTCS instances to Virtual Machines on one physical computer lead to a reduced performance in receiving transport orders.

For the communication between openTCS and the vehicles in Tecnomatix Plant Simulation, we developed a lightweight, text-based protocol. openTCS offers the possibility to develop so-called adapters for special communication methods with vehicles. We used this interface for writing a respective Java-based adapter. Details about the communication contents can be found in the following section.

D. Material Flow Simulation

For simulating the behavior of the transport system with its vehicles, loading/unloading stations, paths and intersections, we use the software Tecnomatix Plant Simulation. This tool is widely used in the industry for simulating production facilities and intralogistic transport systems.

In another research project at the Chair of Materials Handling, Material Flow, Logistics at the Technical University of Munich, an automatic layout generator was developed for Tecnomatix Plant Simulation [6]. By reading in an XML file with the respective information, paths, intersections, load handling stations and different vehicles can be generated. We use this for generating our system from the same XML file we use for import in the openTCS ModelEditor.

For the model of the transport network, the following assumptions have been made:

- Vehicle types are differentiated by the maximum forward and backward velocities, maximum accelerations and minimum curve radius.
- The footprint, i.e. length and width, is the same for every vehicle type.
- Load handling stations, i.e. sources and drains of load carriers, are created via light barriers attached to edges of the transport network. Therefore, if an edge is bidirectional, the load handling stations at this edge can be reached in both driving directions.
- The load handling process itself is modeled by a waiting time of constant duration.

In Figure 5 on the right side, the described elements can be seen in a screenshot of Tecnomatix Plant Simulation. The representation of the same elements in openTCS can be seen in the left half. Here, one of the intersections has been zoomed in to see its four subpoints as well as the in- and outgoing path lines.

To tell the vehicles which route to follow, we use network communication with raw TCP packets. They are exchanged between the Vehicle Driver in openTCS and a "Socket" object in Tecnomatix Plant Simulation. The Socket is configured to be a server socket for TCP packets which means that it can handle several client connections at the same time. The clients are made up by the several instances of the Vehicle Driver in openTCS, as each vehicle gets its own instance for communication there.

The TCP packets contain a string in a special structure: Several sections are separated by semicolons. The first section contains the vehicle's name, the second section contains the message type and the optional third section contains the payload, e.g. a route as an array of node names or a position update. See Table I for the message types from fleet controller to Tecnomatix Plant Simulation and Table II for the message types returned to the fleet controller.

Discrete-event simulation tools like Tecnomatix

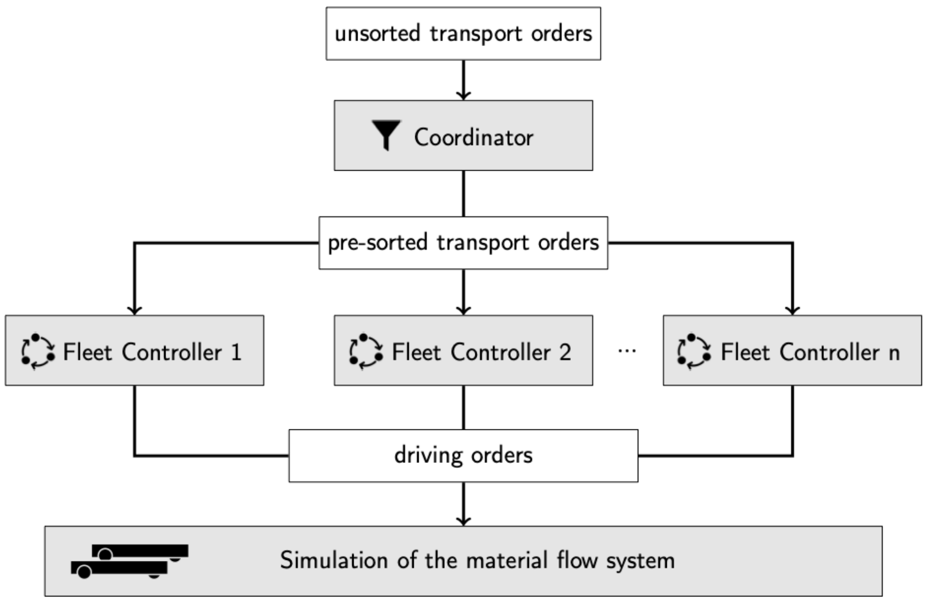
unsorted transport orders
Coordinator
pre-sorted transport orders
Fleet Controller 1
Fleet Controller 2
···
Fleet Controller n
driving orders
Simulation of the material flow system

Figure 4: Architecture of the Software-in-the-Loop system around the system coordinator.

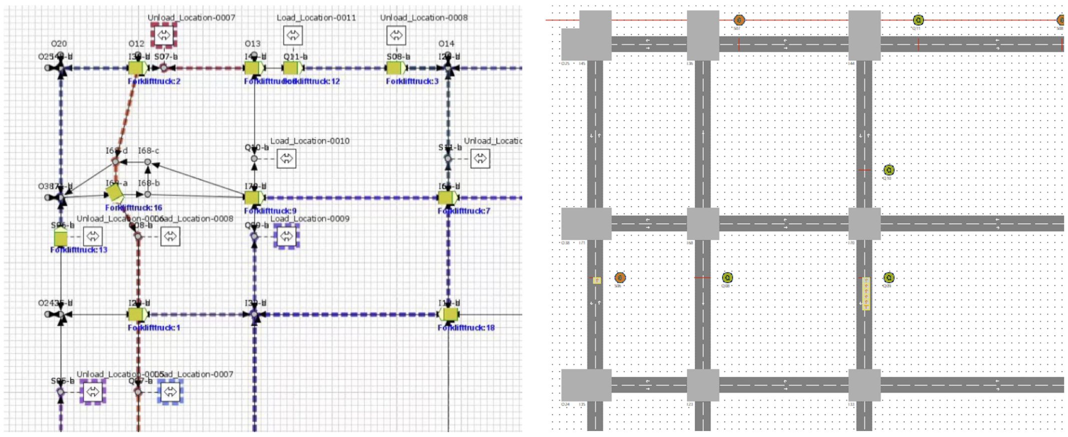

Figure 5: Examplary elements of the transport network, modeled in the fleet controller openTCS (left) and the discrete-event simulation software Tecnomatix Plant Simulation (right).

Table I: Messsage types for the communication from the fleet controller openTCS to the simulated vehicles in Tecnomatix Plant Simulation.

Message type	Parameter
`connect`	vehicle name
`disconnect`	–
`route`	vehicle name + route as array of node names

Table II: Messsage types for the communication from Tecnomatix Plant Simulation to the fleet controller openTCS.

Message type	Parameter
`conn_ack`	–
`disconnect`	–
`position`	vehicle name + node name

Plant Simulation offer the feature of jumping over time intervals where no events occur. This is usually beneficial for speeding up simulations and efficient usage of computing power. In our case, however, the resulting fluctuations in the scale of simulation time would be a disadvantage: The other software parts expect realistic and constant velocities of the vehicles as they incorporate estimations about the future system state into their control commands. Therefore, we used the so-called “1:1” mode of Tecnomatix Plant Simulation that keeps the simulation speed at the constant factor of 1.

E. ERP System Mock-Up

For feeding transport orders into the system coordinator, we need a realistic mock-up of an upstream system. This could be an ERP or ME system. The relevant characteristics of the system mock-up are sending the right information at the right time. The **right in-**

formation contains one or more stations and related actions (loading, unloading, charging etc.). The **right time** means sending transport orders in a frequency that is aligned with the size of the material flow system. Additionally, the frequency should fluctuate in a realistic manner.

We meet these requirements by pre-generating a list of transport orders, each with two stations. At the first station, a loading operation is added and at the second station, an unloading operation is added. This corresponds to the most common form of transport orders. The list contains timestamps relative to the simulation start for the moment they should become active. The interval between two timestamps is chosen according to the size of the transport layout and randomized in an interval of 30 percent around the expectancy value.

The list of transport orders is read in and sent to the coordinator by a separate software module during the runtime of the simulation.

F. Performance Measurement Methodology

An important criterion for a material flow system to be regarded as effective is the timing: Transport orders should be fulfilled as soon as possible after their arrival at the control system. Additionally, they should not take a long time so that the transport system can fulfill more orders per time. And thirdly, deadlines of orders should be met. This sometimes requires prioritizing orders. Other criteria of effectiveness like low energy consumption, few encounters with human workers etc. are less important.

We measure the effectiveness of the coordinative control approach by taking three timestamps of each processed order:

- the moment the coordinator sends it to one of the fleet controllers, T_{send},
- the instant the fleet controller starts the fulfillment of the order by sending the planned route to a vehicle, T_{start},
- the moment the order is finished, T_{finish}.

As openTCS has neither built-in support for pushing events like these to external software nor for a customized logging, we get the timestamps by a polling mechanism in the coordinator. Once a second, the state of all transport orders in each participating fleet controllers is gathered. State changes can thereby be tracked and logged centrally in the coordinator software. The polling method limits the timestamp resolution to 2 seconds which is considered sufficient as the relevant time intervals are in the dimension of one to several minutes.

From the three different time stamps, we can extract three time intervals that are relevant to assess the efficiency of a material flow system:

- the cycle time $t_C = T_{finish} - T_{send}$,
- the processing time $t_P = T_{finish} - T_{start}$,
- the waiting time $t_W = T_{start} - T_{send}$.

In the following, we concentrate on the cycle times, as these allow a judgement of the whole process of order distribution and order fulfilment at the same time.

IV. EVALUATION

The feasability of the overall concept of a software-in-the-loop test bench for a system coordinator shall be proven by a preliminary experiment. In this section, we firstly list the used parameters and secondly present and interpret the results. The proof-of-concept is assumed successful if there is a significant improvement in the results of Configuration 2 of the coordinator in comparison to Configuration 1.

A. Parameters of the Experiment

We use the following parameters for the proof-of-concept:

Layout. The transport network has a square pattern. It consists of 51 intersections and 155 pieces of road (both uni- and bidirectional) connecting them. Along the edges, there are 21 loading and 13 unloading stations, distributed on the layout. The dimensions of the layout are 300 by 150 meters in total. The layout characteristics are derived from a real facility of a German manufacturer of heavy duty vehicles.

Vehicles. There are three types of vehicles used in the material flow system. Each has a transport capacity of one load carrier. All vehicles can handle the same standard type of load carrier. The differences of the vehicle types are their velocity as well as their number in the respective fleet. These data are listed in Table III.

Table III: Vehicle types with differing parameters and their multiplicities.

Vehicle type	**Max. velocity**	**Number**
Forklift	$2\,\mathrm{m\,s^{-1}}$	6
AGV	$1\,\mathrm{m\,s^{-1}}$	6

Order List and Simulation Interval. The simulation interval was chosen to be one hour, which represents one eighth of a shift. In the first 5 minutes of the simulation interval, there are transient effects as the empty vehicles move from the spawning point to the starting points of their first orders. After these effects settle, the simulation can be considered stable since we use a random order list. While this approach is sufficient for comparing different configurations of the system coordinator, it would not be sufficient to make absolute statements about a configuration of the system coordinator with a specific set of vehicle systems and a particular layout. This would require a simulation interval of at least 8 hours and an order list that reflects variations in transportation demand over the course of a shift.

The input order list was set up as described in Section III-E with a waiting time between two orders of 5 to 9 seconds. In total, the list consists of 550 orders.

Tested Coordinator Configurations. The aim of the presented software-in-the-loop test bench is to compare different configurations of the system coordinator described in Section III-A. For the proof-of-concept,

we select two configurations and check, if the respective result data show a reasonable difference:
Configuration 1: In the reference configuration, the system coordinator does not have any intelligence. It has no information about the status of the connected fleets like system usage or vehicle positions and therefore cannot choose a good system to fulfill the orders. Every incoming order is forwarded to a random fleet controller.
Configuration 2: Here, the coordinator knows about the current routes and positions of all vehicles. It feeds them into a time window graph so that it can plan collision-free routes for vehicles to fulfill newly incoming orders. A newly planned route is not added to the time window graph, as the transport order is not directly forwarded to the foreseen vehicle but only to the respective fleet controller. The fleet controller might select another vehicle for the completion of the order than the coordinator. Therefore, the coordinator only stores routes from the fleet controllers in his internal representation. The time window-based preprocessing as well as other algorithms for the coordination with different information needs will be subject to future publications. The basics can be found in [3].

B. Results of the Experiment

Now we take a look at the results of two one-hour simulation runs with the two above-mentioned coordinator configurations. For each run, the same order list was used as input. For the comparative analysis of the two runs, we concentrate on the cycle times and numbers of processed orders.

In Figure 6, the distributions of cycle times are depicted for both simulation variants. We can see a significant difference in the median . Also, the 50 percent interval as well as the 95 percent interval are smaller and lower for Configuration 2. All these characteristics show that Configuration 2 yields lower, less scattered cycle times for transport orders. Another evidence of this are the outliers with the highest one being under 200 seconds for Configuration 2 compared to more than 600 seconds for Configuration 1.

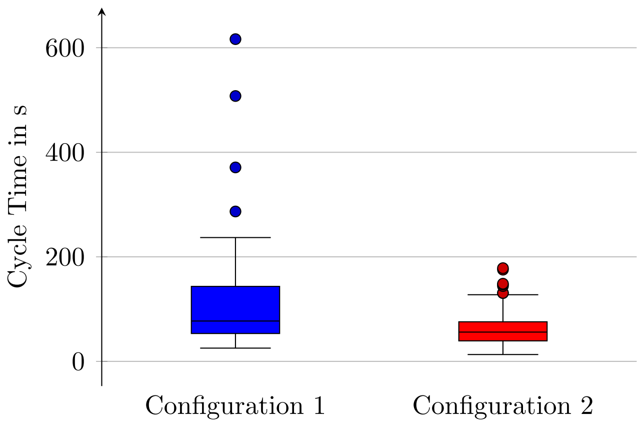

Figure 6: Distribution of cycle times of transport orders for the system coordinator in reference configuration (left) and time window sort algorithm (right).

Looking at the sums of cycle times for both variants (see Figure 7), we realize that the sum is higher for Configuration 2, which has smart routing. Here, it is approximately 5496 seconds compared to approximately 4460 seconds for the reference configuration. Given the same simulation period ($t_{Sim} = 1\,\mathrm{h} = 3\,600\,\mathrm{s}$) for both configurations and the higher number of fulfilled orders in Configuration 2 (88 vs. 31), we can suppose that the vehicle usage ρ is higher for Configuration 2. For calculating ρ, we need the simulation period t_{Sim}, the number of vehicles $n_{Vehicles}$ and the sum of processing times $\sum t_P$:

$$\rho = \frac{\sum t_P}{n_{Vehicles} * t_{Sim}}$$

With the sums of the processing times of all orders being $\sum t_P|_{Configuration\ 1} = 2\,052.71\,\mathrm{s}$ for Configuration 1 and $\sum t_P|_{Configuration\ 2} = 5\,314.11\,\mathrm{s}$ for Configuration 2, we can calculate the values for ρ:

$$\rho_{Configuration\ 1} = \frac{2\,052.71\,\mathrm{s}}{8 * 3\,600\,\mathrm{s}} = 7.13\,\%$$
$$\rho_{Configuration\ 2} = \frac{5\,314.1\,\mathrm{s}}{8 * 3\,600\,\mathrm{s}} = 18.5\,\%$$

The higher vehicle usage is also reflected by the difference in the number of fulfilled transport orders. One reason for this could be the more frequent occurrence of deadlocks between vehicles in Configuration 1 (see the outliers in Figure 6). These were resolved manually. Deadlocks are more likely to occur when vehicles follow routes that have not been aligned with the routes of other vehicles, like in Configuration 1.

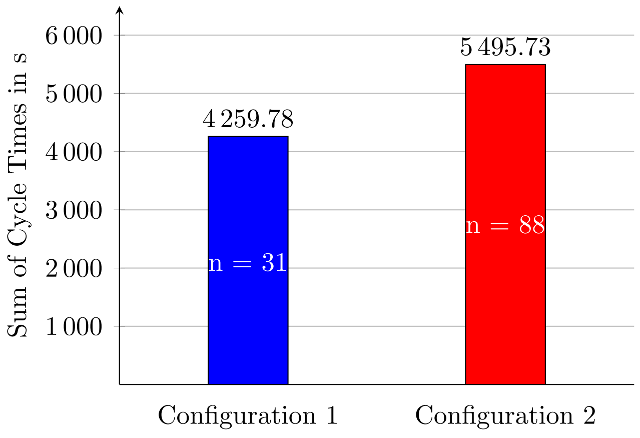

Figure 7: Sums and absolute numbers of cycle times of transport orders for the system coordinator in reference configuration (left) and time window sort algorithm (right).

In conclusion, the results of the experiment show that intelligent order distribution by the coordinator using a time window-based algorithm is superior to random order distribution. The working hypothesis is justified and we therefore assume that the overall software-in-the-loop test bench is valid for testing coordinative order distribution in material flow systems.

V. CONCLUSIONS AND OUTLOOK

We have shown in this article that simulation of multiple independent vehicle fleets in a common material flow system can be achieved by combining several software packages. We were also able to show that the coordinative control approach for intralogistic transport systems works.

The results of the two test runs are plausible relative to each other. However, the absolute values are questionable, since significantly higher utilization rates are achieved in reality.

Current Drawbacks. First, the deadlock avoidance in the Tecnomatix Plant Simulation model should be improved. Furthermore, the edges in the transport network in openTCS have zero capacity, so there is significant difference between the vehicle positions in Tecnomatix Plant Simulation and openTCS during simulation runs. As a result, openTCS cannot always make good decisions about which vehicle should execute which transport order. Also, orders are sometimes registered as completed too late and thus vehicles that are actually free again are not considered by openTCS.

The pull principle in acquiring the timestamps limits their accuracy and leads to a high data traffic at the kernel, in the local network and in the coordinator. An investigation whether this amount of communication has a negative impact on the simulation is still pending.

Outlook. With regard to the asynchrony of the vehicle positions in openTCS and Tecnomatix Plant Simulation, the goal is to discretely subdivide the edges in openTCS. This should allow several vehicles to drive one after the other on one edge in openTCS and thus block the intersections for a shorter time. This can be done by adapting the XML layout being imported into openTCS.

Finally, a detailed simulation study is to be carried out after the aforementioned improvements. In this, several configurations of the system coordinator with different interfaces and distribution mechanisms are to be compared. This should ultimately enable a statement to be made on the optimum interface design of the system coordinator.

ACKNOWLEDGEMENTS

The results presented in this article were developed within the research project “EPoSysKo – Entwicklung und Potentialanalyse eines Systemkoordinators für den typübergreifenden Einsatz von Flurförderzeugen”. The research project was carried out under the code 20918 N on behalf of the Research Association Bundesvereinigung Logistik e.V. (BVL) and was funded by the German Federal Ministry of Economics and Climate Protection via the German Federation of Industrial Research Associations “Otto von Guericke” e.V. (AiF).

REFERENCES

[1] Changsheng Ai, Chao Zhuang, Li Song, Chunxia Yang, and Yungang Guo. AGV scheduling system based on MQTT protocol. In *2021 4th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*. IEEE, 2021.

[2] Maximilian Berndt, Dennis Krummacker, Christoph Fischer, and Hans Dieter Schotten. Centralized robotic fleet coordination and control. *Mobile Communication - Technologies and Applications; 25th ITG-Symposium*, 2021.

[3] Johannes Fottner and Florian Rothmeyer. *EPoSysKo – Entwicklung und Potentialanalyse eines Systemkoordinators für den typübergreifenden Einsatz von Flurförderzeugen: Forschungsbericht*. Lehrstuhl für Fördertechnik Materialfluss Logistik, München, 2022.

[4] International Federation of Robotics. World robotics 2022. `https://ifr.org/downloads/press2018/2022_WR_extended_version.pdf`, 2022.

[5] Thomas Lienert. *Methodik zur simulationsbasierten Durchsatzanalyse FTF-basierter Kommissioniersysteme.* PhD thesis, Technische Universität München, 2021.

[6] Slaheddine Mestiri, Jazib Jamil, and Johannes Fottner. A flexible and generic simulation model for in-bound transport systems. pages 85–90.

[7] The openTCS developers. openTCS: User’s guide: Version 5.6.0. `https://www.opentcs.org/docs/5.6/user/opentcs-users-guide.html`.

[8] VDA. Schnittstelle zur Kommunikation zwischen fahrerlosen Transportfahrzeugen (FTF) und einer Leitsteuerung. `https://en.vda.de/dam/vda/publications/2020/Produktion-und-Logistik/VDA5050_DE_V1.1.pdf`, 2020.

[9] VDI-Gesellschaft Produktion und Logistik. Kompatibilität von Fahrerlosen Transportsystemen (FTS) - Leitsteuerung für FTS, 2005.

[10] Andor Bálint Viharos and István Németh. Simulation and scheduling of AGV based robotic assembly systems. *IFAC-PapersOnLine*, 51(11):1415–1420, 2018.

[11] Matthias Wissing, Frank Kuenemund, Daniel Hess, and Christof Roehrig. Hybrid navigation system for mecanum based omnidirectional automated guided vehicles. In *ISR/Robotik 2014; 41st International Symposium on Robotics*, pages 1–6, 2014.

FLORIAN ROTHMEYER has been working as a research associate at the Chair of Materials Handling, Material Flow and Logistics, Technical University of Munich, since 2020. With a background in Mechatronics and Information Technology, he works on improving mobile robots for intralogistics from the vehicle hardware up to the fleet controller. His research focus is the orchestration of large, heterogenous fleets of mobile robots in conjunction with manually driven vehicles.

JOHANNES FOTTNER received a Dr.-Ing. degree in mechanical engineering from the Technical University of Munich (TUM), Munich, Germany in 2002. From 2002 to 2016, Johannes Fottner acted in different managing functions in the materials handling sector and gained expertise in the development, implementation and operation of automated logistics and production systems composed of different modules. Since 2016, he has been the head of the Chair of Materials Handling, Material Flow, Logistics at the TUM. His current research interests include investigating innovative technical solutions and system approaches for optimizing logistical processes.