Computers in Industry 64 (2013) 663–677

Contents lists available at SciVerse ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

A generic material flow control model applied in two industrial sectors

S.W.A. Haneyah [*], P.C. Schuur [1], J.M.J. Schutten [2], W.H.M. Zijm [3]

Industrial Engineering and Business Information Systems (IEBIS), School of Management and Governance, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands

ARTICLE INFO

Article history:
Received 10 May 2012
Received in revised form 27 January 2013
Accepted 19 March 2013
Available online 12 April 2013

Keywords:
Automated material handling systems (AMHSs)
Real-time scheduling
Generic control architecture
Baggage Handling
Distribution

ABSTRACT

This paper addresses the problem of generic planning and control of automated material handling systems (AMHSs). We build upon previous work to provide a proof of concept for generic control of AMHSs in different domains. We present a generic control architecture for AMHSs, and apply this architecture to a material flow model with storage and sorter systems. We set up our model to be applicable to AMHSs in two different industrial sectors: Baggage Handling and Distribution. We report on performance indicators and analyze how far we can control the two industries generically in terms of software implementation. To this end, we present an impressive degree of 84% commonality in the control software code. Moreover, we highlight deviations from the generic control and give insight to control procedures that deviate from the generic code. A generic architecture that optimally exploits synergy between the different market sectors may reduce design time and costs considerably for system suppliers acting in both industries, while finding a common ground to model AMHSs in these different sectors also forms a scientific challenge.

1. Introduction

In this paper, we introduce a generic control architecture and apply it to two industrial sectors that are very similar in terms of the hardware used, but very different in terms of the operational environment, i.e., Baggage Handling and Distribution, respectively (see Section 1.2). In both sectors, an *Automated Storage and Retrieval System* (*ASRS*) is a vital element. The model reflects the first building block of a concept control architecture we proposed in (see [14]), which is in line with the specified design requirements: flexibility, modularity, scalability and robustness. The functionality of the control architecture depends on the decision making processes involved, which have to meet primary functional requirements, in the first place throughput and response time, and in addition several secondary requirements, e.g., workload balancing and congestion avoidance (see [14], for more details). In concrete terms, the purpose of our model is to be a proof of concept for a generic control system for AMHSs in different domains.

In this section, we briefly explain the context of generic planning and control of AMHSs (Section 1.1), and the processes in the industrial sectors, i.e., Baggage Handling and Distribution (Section 2.1), in which we apply the generic material flow control model. This research is heavily motivated by our experience with a major global company supplying AMHSs in the market sectors we consider.

1.1. Project context

In an earlier paper [14], we focus on the problem of generic planning and control of Automated Material Handling Systems (AMHSs), in three different industrial sectors: Baggage Handling, Distribution, and Parcel and Postal. AMHSs are in general complex installations that comprise various processes, such as inbound, storage, batching, sorting, picking, and outbound processes. Typical performance indicators for these systems concern throughput, lead time, and reliability. Although AMHSs in these different sectors share a similar hardware infrastructure, their planning and control remains highly customized and project-specific, which is a disadvantage from both the systems' supplier and the systems' user point of view. From a user perspective, the environment and user requirements of systems may vary over time, requiring the adaptation of the planning and control procedures. From a supplier perspective, design time and costs can be considerably reduced using a comprehensive planning and control architecture that exploits synergy between the distinct market sectors, and yet is flexible with respect to changing

* Corresponding author. Tel.: +31 53 489 3603; fax: +31 53 489 2159.
E-mail addresses: s.w.a.haneyah@utwente.nl, s.w.a.haneyah@alumnus.utwente.nl (S.W.A. Haneyah), p.c.schuur@utwente.nl (P.C. Schuur), m.schutten@utwente.nl (J.M.J. Schutten), w.h.m.zijm@utwente.nl (W.H.M. Zijm).
[1] Tel.: +31 53 489 3658; fax: +31 53 489 2159.
[2] Tel.: +31 53 489 4676; fax: +31 53 489 2159.
[3] Tel.: +31 53 489 3912; fax: +31 53 489 2159.

0166-3615/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.compind.2013.03.007

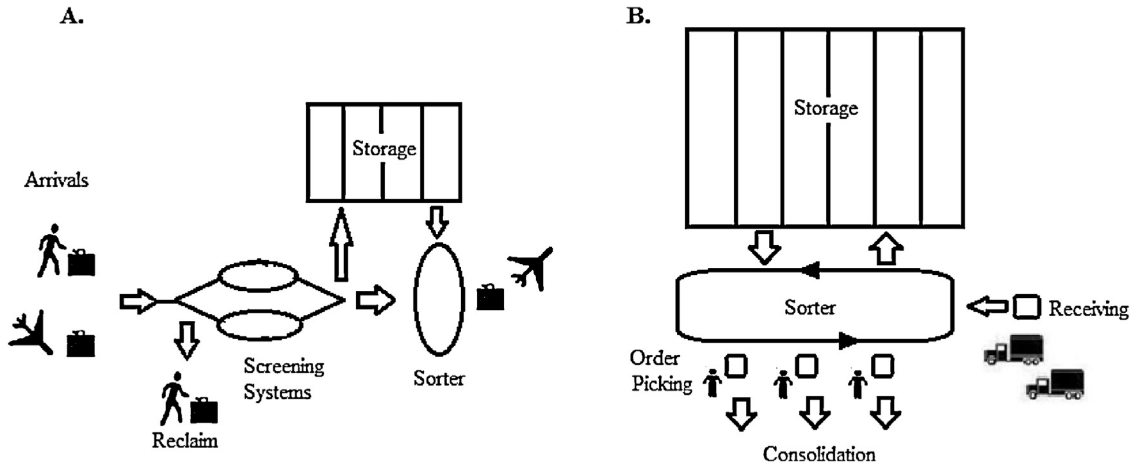


Fig. 1. Basic overview of a baggage handling and a distribution system.

business parameters and objectives. From a scientific perspective finally, we address the challenge of modeling AMHSs in entirely different industrial sectors generically, and provide more insight into the similarities and the differences of these systems.

1.2. Processes description

Fig. 1A presents a high level overview of a baggage handling system (BHS) in airports. Bags arriving at a BHS originate either from check-in desks, or from transfer flights. Bags arriving at their final destination are directed to baggage reclaim belts. Transfer bags, on the contrary, have to go through screening systems and then they either go to *early bag storage* (if their flight is not yet open for loading), or directly to the sorter where bags are sorted to flights loading for departure and so leave the BHS.

Fig. 1B presents a high level overview of a distribution system. Arrivals, e.g., product totes, appear at the receiving station, and are then handled by the sorting loop. Totes are first stored in the storage area and can be used later to satisfy customer orders at the order picking stations. Totes used at the pick stations may return to the storage area (via the sorting loop) if their content is not completely consumed. Consolidation and shipping of customer orders occur after order picking.

In our generic model of this paper, we focus mainly on the sorter system and the storage system. For more details about the process description in these two market sectors, and for a detailed comparison between AMHSs in different industrial sectors we refer to [14].

These two industrial sectors are the application areas for our material flow control model for two reasons. First, these both represent important sectors for the materials handling industry, next to parcel and postal sorting systems. The latter systems are excluded from this study as they do not entail the storage function, which is a main element in the model at hand. Second and more important, we observe that these two industrial sectors are very similar in terms of the equipment used (e.g., ASRSs, conveyors as the mean of transport, sorting systems), but are very different in terms of the operational environment, and as a result these sectors use entirely different key performance indicators (KPIs). In an airport a flight has to depart on time and does not wait for a late bag (a late bag will miss the flight). On the other hand, in a distribution center order lines are processed based on the progress of work rather than strict time schedules. Therefore, an order picking station waits until required totes arrive, even if they are delayed. Also, in a baggage system, item integrity is prevailed, whereas in distribution systems the contents of a storage bin may be consumed only partially.

In these different operational environments, it is not possible to define a single key performance indicator (KPI) for both industrial sectors. There is, however, in both industries a clear notion of what defines a better solution. Airports are mainly interested in one aspect of baggage handling systems: the number of bags that do not make their flight as a result of a failing handling system, also known as the *irregularity rate* (measured per 1000 bags). In Distribution, however, the focus is on *throughput*. Throughput is generally defined as the number of order lines processed per hour.

The paper proceeds as follows: in Section 2, we define our theoretical framework and conduct a literature review in a search for generic control methods for AMHSs. Section 3 is devoted to the modeling and analysis with regard to the proposed control architecture, decision-making processes, and material flow control model. Section 4 covers the implementation of the proposed control architecture on the generic material flow model, in terms of experimental results and the realization of our generic software implementation target. Finally, in Section 5, we present conclusions.

2. Theory

In this section, we first present a theoretical framework in order to define the scope of our study, and to position the studies in the literature review in a certain framework. In Section 2.1 we define this theoretical framework. Thereafter, in Section 2.2 we discuss the main literature contributions and position these studies using the reference framework of Section 2.1.

2.1. Theoretical framework

The theoretical framework has two dimensions. The first one is concerned with the decision-making framework, while the second is concerned with the control structure, which is basically the control architecture on which decision-making processes are mapped. Our decision-making framework builds upon the findings of an earlier paper [14], and upon established theories in the temporal decomposition of planning, scheduling and control processes (see [3,15,32]). We propose a decision-making framework with three hierarchical levels of control (see Fig. 2), as follows:

- *Planning*: Planning processes are those requiring a global view of the system regardless of the system size. This is the control level that interacts with the outer environment, e.g., customer orders, plane schedule changes.

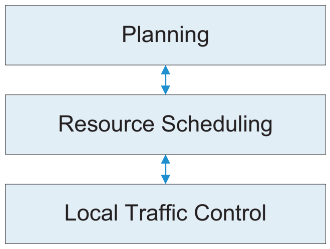


Fig. 2. Levels of control.

- *Scheduling*: Given a set of assigned tasks, this level addresses the problem: when and in what sequence to execute these tasks.
- *Local traffic control*: this level entails algorithms or routing rules executed within defined boundaries of the physical system. There is minimal interaction with other areas in the system, and mostly the aim is local optimization where no global view is needed.

We focus on high levels of control that deal with decision-making functions, and not on implementation issues, e.g., configuration of hardware elements, equipment instructions, and conveyor movements. Therefore, due to our functional rather than software implementation focus, we may exclude some basic principles of collaborative control theory. For example, Conflict and Error Diagnostics and Prognostics (CEDP) is a basic principle that is often studied in literature, e.g., in [6]. However, CEDP focuses on software-related issues, i.e., the prediction and detection of errors in the software code, which is beyond our scope. As a matter of fact, we believe that on this level (e.g., machine interfaces, equipment control), standardization independent of specific applications is already the rule rather than the exception. Therefore, we would like to explore whether a similar standardization may be achieved at higher, more abstract, decision making levels. For examples of studies dealing with low levels of control and configurability, we refer to Alsafi and Vyatkin [1] who present a methodology to integrate the high level planning with low level control of a mechatronic system, and Furmans et al. [9] who propose a plug-and-work MHS.

The second dimension of our theoretical framework, i.e., control structures, builds upon the four basic forms of control that have been suggested in the literature (explained in [14]). We provide a description based on [8], who review the evolution of control architectures grouped in the major four forms of control, as follows (see Fig. 3, where control units are represented by squares and resources by circles):

1. Centralized form: here a central control unit performs all planning and control functions for all resources in the system. Moreover, it uses a global database that contains all types of detailed information about the system.
2. Proper hierarchical form: in this form, there are multiple control units, and a rigid master–slave relation between decision-making levels. The control unit in an upper hierarchy acts as a supervisor for resources in the subordinate level. Decisions made by the supervisor have an aggregate view on the system, and do not prescribe detailed low level actions. Subordinate control units have to comply with tasks imposed by controls in the upper level, but as tasks are delegated, subordinates make more detailed decisions for their actions.
3. Modified hierarchical form: this form evolved in order to deal with some shortcomings in the proper hierarchical form, mainly the rigid master–slave relationship. It differs from the proper hierarchical form primarily through the degree of autonomy of subordinates. In the modified hierarchical form there is some degree of coordination among subordinates on the same hierarchical level.
4. Heterarchical form: this form is the extreme of decentralized control, which became popular recently. An example is a multi-agent system (MAS). In this form, control structures have distributed locally autonomous entities. These entities communicate with each other to make decisions in cooperation. The master–slave relationship is totally abandoned and not just loosened as in the modified hierarchical form.

As a final remark, we have to make a distinction between our focus on *control architectures* (such as described by Dilts et al. [8]), and *software architectures*, because a decentralized control architecture can be implemented, in principle, by a monolithic software architecture and vice versa. However, advantages and disadvantages of centralization versus decentralization in both domains run in parallel to each other and are often mixed up.

2.2. Literature review

In this section, we list studies that are relevant to planning and control of AMHSs in general, and in Baggage Handling and Distribution in particular. We make an attempt to classify the reviewed studies based on the framework for the basic forms of control (Section 2.1).

2.2.1. Centralized control

Tařau et al. [26] study route control in BHSs. They compare centralized and decentralized route choice in BHSs, particularly in systems using *Destination Coded Vehicles* (*DCVs*) as a transport mechanism. They implement centralized control approaches, but find them computationally expensive and not robust. Furthermore, they develop decentralized control rules for *Merge* and *Divert* switches, where each switch has its own controller.

Mo et al. [22] study flow diversion to multiple paths in integrated automatic shipment handling systems. The authors take a network optimization perspective and formulate a nonlinear multi-commodity flow problem. They develop a mathematical programming model to propose routing strategies with the objective of minimizing the total shipment travel time in the system. However, they do not apply their theoretical framework to a business case, and they make assumptions that may not hold in many practical settings. For example, they assume independent waiting times at different pieces of equipment, and do not include time constraints for special shipments.

Zimran [33] presents a commercial generic controller for material handling systems. His design is mostly based on hardware and software linkages and communication. The routing decision

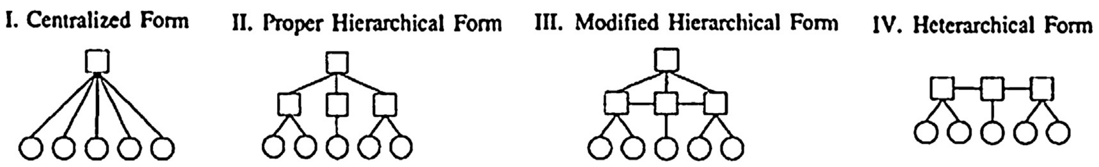


Fig. 3. Evolution of control architectures [8].

function is supported by tree graph algorithms. Tree graphs have only one path between every pair of origin and destination. These tree graphs change while the system is running (based on system state), by adding or removing arcs. Since the algorithm is computationally expensive, simpler algorithms are used for low level controllers.

2.2.2. Hierarchical control

The concept of *Cooperation Requirements Planning* (*CRP*) is a hierarchical decision-making strategy that stems from collaborative control theory. Rajan and Nof [24] define CRP as "the process of generating a consistent and coordinated global execution plan for a set of tasks to be completed by a multi-machine system based on the task cooperation requirements and interactions." CRP is divided into two steps. The first step (CRP I) generates the *cooperation requirements matrix* whose elements represent the capabilities of machine sets for processing the tasks. CRP I also generates processing constraints. Next, the second step (CRP II) determines the assignment of tasks to machine sets for processing. These two steps may include advanced search algorithms to generate plans and to make assignments. In general, CRP is unnecessarily complicated for our AMHSs control problem. It is more adequate for a manufacturing environment such as the robots and machine cells application presented by Rajan and Nof [24]. In such environments, it is challenging to deal with jobs that need several processing tasks, which are not standardized. On the contrary, in the AMHSs we study, items follow standardized routes and processes, but the challenge lies in the control and balance of material flows within the systems.

Amato et al. [2] state that control systems of warehouses have three main hierarchical levels: a planning level, a management level, and a handling level. The authors introduce the *Optimizer System* as a new level to bridge the gap between planning/management and shop floor control systems by improving the realization of decisions by handling devices such as the cranes and a shuttle handling device.

In Baggage Handling, Tařau et al. [27] address hierarchical control for route choice. To this end, they design a control architecture with three levels of hierarchy: network controller, switch controller, and DCV controller. In the same study, they examine multi-agent systems, but find them hard to implement due to the extensive communication required between the agents. In general, Tařau et al. [26,27] focus on BHSs, and only on routing by controlling switches within BHSs, but they do not consider the storage operation.

2.2.3. Modified hierarchical control

Kim et al. [19] propose a hybrid scheduling and control architecture for warehouse management, mainly for order picking. We can classify their architecture as modified hierarchical, although they implement it using multi-agents software. In their architecture, they have three hierarchical levels of control: high level optimizer agent, medium level guide agent, and low level agents, which have some degree of autonomy. The fact that this architecture is tailored to order picking in a warehouse limits it applicability as a generic control architecture for AMHSs.

2.2.4. Heterarchical control

As a matter of fact, heterarchical forms of control are the recent trend in research. Babiceanu et al. [5] present a framework for the control of AMHSs as part of the so-called holonic manufacturing approach. Holons are units that act as parts and as wholes at the same time, meaning that they have a high degree of autonomy but operate as part of a more general system. Therefore, holons have two main properties: autonomy in making decisions, and cooperation with other holons for mutually acceptable plans. The authors state that from the significant number of papers in the area of agent-based and holonic manufacturing, only very few consider material handling problems. They present a case study focusing on a material handling system.

Van Brussel et al. [28] present a reference architecture for holonic manufacturing systems. Their architecture has 3 main holons:

- Product holon: represents a product model of a product type, which basically acts as an information server to other agents.
- Resource holon: represents a production resource in the system.
- Order holon: represents a task with requirements and a due date. It manages a physical product being produced.

In addition, staff holons are optional holons that can aid other holons in decision-making. An example is a central scheduling unit. The architecture is called PROSA, which stands for Product–Resource–Order–Staff–Architecture. PROSA focuses primarily on manufacturing operations rather than transport operations. In this paper, however, we do not aim for an architecture that is generic for AMHSs and for manufacturing systems; we focus solely on AMHSs and the operations within the market sectors we analyze. The complexity of decision-making in the AMHSs we study is less than that for a flexible manufacturing cell and, more important, is of a different nature. The PROSA is an example of a completely heterarchical control approach, whereas we will opt, for good reasons, for another form of control (see Section 3).

The holonic paradigm is similar to the agent paradigm in many aspects, but there are some differences. Giret and Botti [10] conduct a thorough study to provide a comprehensive comparison of holons and agents. Their main conclusion is that a holon is a special case of an agent. A holonic system is a manufacturing-specific approach for distributed intelligent control. On the other hand, a multi-agent system is a broad software approach, where one of its uses is distributed intelligent control. For more details, we refer to Giret and Botti [10]. However, we note that holonic systems are heterarchical in the context of the systems we address in this paper, but they may have hierarchical characteristics when applied to other types of systems that are beyond the scope of this paper.

Vrba and Mařík [30] focus on software implementation and the use of simulation in agent-based control systems. In their control architecture, they use a basic set of agents for conveyor-based transportation: work cell, diverter, and conveyor belt. In this work, we find useful control mechanisms such as the dynamic routing tables used by the diverters. However, diverters are a subject of our future studies and are not included in the control model of this paper. We stress that the main objective of our research is to propose a generic control architecture for AMHSs that is applicable in different market sectors, where not every element within this architecture is necessarily a novel application.

Lau and Woo [20] develop an agent-based dynamic routing strategy for AMHSs. They emphasize that existing routing strategies in theory often use static routing information based on shortest path, least utilization, round-robin assignments, etc. In their study, they map the AMHS to a network with node agents connected by unidirectional links. Control points of a network of AMHS components are modeled as cooperating node agents. To make routing decisions, they define the best route in terms of: cycle time of material, workload balancing, and degree of tolerance to unexpected events. In their architecture, each agent is responsible for its zone of coverage. They implement their architecture in a simulation environment of a DC. The authors outline a generic classification of routing strategies and classify their approach as *distributed real-time state-dependent*.

Johnstone et al. [18] study status-based routing in Baggage Handling. In their approach, the status of the bag determines its processing requirements, and triggers computation of the route to be followed depending on the states of required resources ahead. The authors study two main algorithms, the first one based on learning agents, while the second uses a graph representation of the network to find all possible routes at switches via Dijkstra's shortest path algorithm. They find learning agents more efficient in larger systems, as they make use of information from operations performed on the bag upstream. With this information, they limit the possible routing options downstream.

Hallenborg and Demazeau [13] use multi-agent technology in a BHS to construct generic software components to replace traditional system-specific centralized control software. In their approach, when the bag enters the system, the first agent on the route can make an agreement with all agents on the route to the bag's destination. However, it is also possible to make an agreement only with the next agent on the route. This raises the distinction between routing by static shortest path, or routing on the way.

Some of the advanced control designs generate forecasts in order to prevent congestions, and to facilitate proactive rather than reactive decisions. Studies in this context include Hadeli et al. [12] who present a control architecture that is a combination of the PROSA and concepts inspired by ant colony coordination mechanisms. Weyns et al. [31] use delegate MASs, inspired by food foraging in ant colonies, to anticipate road conditions to make routing decisions. Claes et al. [7] present an MAS for anticipatory vehicle routing, which allows directing vehicle routes by accounting for traffic forecast information. Finally, Parunak [23] presents the concept of swarming agents that interact through digital pheromones. However, note that we focus on internal transport, as distinguished from external transport that is dealt with in these studies. Moreover, we employ other anticipation techniques that take precautions in order not to create congestions, and in order to maintain a balanced material flow in the system. Section 3 further describes control approaches that are related to the model we present in this paper.

Some simulation-based studies in the area of AMHSs are worth to be mentioned. [21] present a modular simulation approach for the evaluation of AMHSs. Babiceanu and Chen [4] use simulation to justify the use of a decentralized agent-based approach in materials handling and assess its performance compared to conventional scheduling systems. Jahangirian et al. [17] conduct a broad review of simulation studies in manufacturing. A trend they notice concerns the increasing interest in hybrid modeling as an approach to cope with complex enterprise-wide systems. Hunter [16] presents a model evolution analysis for simulating AMHSs. Finally, we mention Van den Berg [29], Rouwenhorst et al. [25], and Gu et al. [11] as useful literature reviews in the Distribution and warehousing area.

2.2.5. *Conclusion*

As a general remark, there are few studies that attempt to build a generic control architecture for MHSs operating in different market sectors. From the studies we reviewed, we observe that a control architecture normally targets a specific sector, or deals with material handling as part of a manufacturing environment. From our point of view, the most relevant study is the holonic architecture proposed by Babiceanu et al. [5]. Although this architecture is based on a manufacturing system, it does suggest a framework for material handling. However, the AMHSs in the sectors we address are far more complex and diverse than the AMHS modeled by Babiceanu et al. [5]. We conclude that their study misses an in-depth treatment of practical requirements of complex AMHSs. Moreover, the authors focus on the design aspect, but do not show how decision-making processes can be employed to achieve functional requirements. In general, many authors favor distributed control when dealing with complex systems.

From the studies we reviewed, we observe that a control architecture is initially designed and then applied to some sector, often to a distribution center. For Baggage Handling, there are few studies on control architectures. Most of the studies focus on route planning through divert and merge switches, and do not take the storage operation into account. On the other hand, the relatively abundant number of studies on warehousing systems emphasize either the design aspects, or throughput optimization of the system through the use of advanced algorithms for warehousing activities such as storage and retrieval sequencing, and order pick concepts. From our intensive experience with industry we however learned that other requirements are necessary to make the control architecture applicable in a practical setting. For example, experts from industry value a robust control architecture that provides satisfactory solutions higher than an architecture that provides near optimal solutions but is less robust.

3. Modeling and analysis

3.1. *Proposed control architecture*

In a previous paper [14], we built upon the collaboration with the MHS industry and upon the basic forms of control outlined above, to evaluate the alternative forms of control. We excluded the centralized approach for different reasons, including the computation time, difficulties in dealing with information flowing in real-time, difficulties in dealing with disruptions in material flow, and the complex software structure that does not serve requirements of being generic, modular, robust, and flexible.

The centralized approach is one extreme of decision making; the other extreme is purely decentralized decision-making embodied by the heterarchical approach. The literature review suggests that the heterarchical approach is the best to support objectives such as modularity, genericity, and robustness. In the heterarchical form of control the modular architecture can be composed by configuring the interfaces between different software components. However, we also indicated that a pure heterarchical form of control results in a cooperative approach to global decision making, where a main concern is the extent of deviation from the optimal solution. Another concern is the loss of higher level coordination that may be necessary in some cases, e.g., planning orders. Note that in our generic control problem, decisions made within AMHSs are not all at the same level. In particular, when considering different market sectors, we find global decisions that impact the overall performance of the system, while others are local decisions with limited global impact.

Indeed, distributed control is beneficial when dealing with complex systems. However, we emphasize that distributed control means having *decisions made at the right level*, and thus it can be realized with other forms of control, e.g., the modified hierarchical form, to at least link different levels.

Based on the aforementioned points and observations in industry, we proposed a concept control architecture that involves hierarchical control and also a certain degree of intelligence and freedom of controllers at different control levels [14].

Fig. 4 shows the proposed control architecture, which has two levels of control, each with several control units. At the planning level, there are planning control units, referred to as *planners*, which have an aggregate view of the system and are not directly connected to system resources. On the other hand, at the scheduling level, there are scheduling control units, referred to as *schedulers*, which are directly connected to system resources,

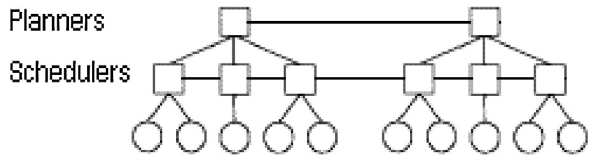


Fig. 4. Control architecture scheme.

being workstations or transport resources. Planners communicate with each other, and assign tasks to subordinate schedulers. Schedulers also communicate with each other to schedule the assigned tasks, and report to higher level planners. The proposed control architecture has a certain degree of hierarchy in combination with flexible decision making the subordinates. Therefore, it is a variant of the modified hierarchical form of control (see Fig. 3). Local traffic control is implemented at a low level of control, and is responsible for local decision on transport and movements, e.g., the movement of a crane within its aisle, or prioritizing the movement of items on a conveyor junction. Local traffic rules and algorithms can be executed by schedulers, or implemented at a PLC level (programmable logic controller). However, in this paper we focus on planning and scheduling processes as they define the control architecture. Local traffic decisions are dealt with either as a black box, or using simple rules. We dedicate other studies to local traffic control algorithms.

Having defined the basic structure of our control, we next allocate decision-making activities to the different levels of control and to the different controllers. In doing so, we follow two main principles:

- Any decision-making activity is to be performed at the lowest possible control level and with the narrowest possible scope. Possible means that no direct deterioration in system performance is expected due to making the decision locally and with a relatively narrow scope. However, this principle may be violated if synergy in control among different sectors requires that.
- If an operation with certain characteristics is defined as a scheduling operation, another operation with the same characteristics but on a wider scale due to, e.g., system size, may become a more complex scheduling operation, but does not become a planning operation.

There are two main planners we incorporate in our control architecture:

- *Build planner*: responsible for the build area, i.e., workstations. In Distribution, this means planning the order picking process, whereas in Baggage Handling this means planning the *make-up* of flights, i.e., gathering the baggage belonging to the flight at the right make-up point(s). This is a planner as it requires a global view on system information, schedules, and the build area. Moreover, it results in assigning work to system resources (see our definition of the planning level).
- *Storage planner*: this controller is responsible for the storage area, i.e., the ASRS consisting of cranes and storage aisles. The same arguments as with the build planner hold for this controller to be a planner, where the global view necessary is on the ASRS and operating cranes.

In Section 3.2 we present the system model. Then, we define the controllers and decision-making processes involved (Section 3.3).

3.2. System model

Based on two reference AMHSs we analyzed in two different sectors (Baggage Handling and Distribution; see Fig. 5a and b), we construct a model of a generic AMHS (see Fig. 5c). The generic system entails an ASRS with aisles and cranes, a conveyor in loop configuration as a sorter system, and inbound and outbound conveyors connecting the cranes and workstations to the loop. However, we do not display a comprehensive AMHS, i.e., in a typical baggage handling system there are baggage screening systems, entailing other loop conveyors, and clusters of screening machines. These systems are upstream the ASRS and the main sorter system, which we address in a future study.

A workstation is a generic term that refers to an order picking station in a distribution system. For a baggage handling system, the workstation refers to a *lateral* (a type of outfeed conveyor where baggage for a certain flight is collected).

Transport stock unit (*TSU*) is also a generic term, which we use to refer to different types of items that can be transported in the system, e.g., bags or totes. In Baggage Handling, an arriving TSU goes directly to the sorter system if its flight is open for make-up on one or more workstations; otherwise it is diverted to the ASRS. When the make-up of a flight is 'open,' relevant bags are released from the ASRS to the workstation(s) assigned to handle the baggage for this flight.

In Distribution, arriving TSUs are always stored first, so they never go to workstations directly. TSUs are *full* when they enter the system. However, when a TSU is used for an order, some *SKUs* (*stock keeping units*) are picked from it at a workstation, and then if the TSU is not completely consumed, it becomes a *broken* TSU that has to go back from the workstation to the ASRS.

We highlight that in Distribution arriving TSUs go to the ASRS via the main sorter system (see Fig. 5b), whereas in Baggage Handling arrivals enter the ASRS via another conveyor-based route (see Fig. 5a). The main sorter system only handles the outflow baggage from the ASRS to the workstations. In order to apply generic control methods, we propose the generic AMHS model (Fig. 5c), in which we do not model the transport route leading arriving TSUs (in Baggage Handling) to the ASRS in detail. We model this route in aggregate terms because it is not critical from a control point of view and it does not exist in the distribution system. However, we apply the generic control elements to both industrial sectors by parameterizing the generic AMHS model to simulate distinct sectors. For example, the conveyor route from the ASRS to the sorter in Fig. 5a is incorporated in Fig. 5c by longer travel times on the sorting loop from the ASRS to workstations, where travel times are configurable parameters in a *travel times matrix* (see Section 4.1) in the generic model. The recirculation time on the sorting loop follows the actual travel times (from practice) for each industrial sector. Moreover, when the generic AMHS is configured as a distribution system, then arriving TSUs proceed to the ASRS via the sorting loop and do not use the other (Baggage Handling) route from the divert to the ASRS. In this case, the generic model reduces to the distribution system model (Fig. 5b).

In this paper, we focus on planning the flow of TSUs into the ASRS and out of it to the workstations via the main sorting loop. To this end, we model the following system components:

- *Storage aisles*: we model storage aisles in aggregate terms, i.e., we do not model the exact storage locations within an aisle.
- *Cranes*: cranes are modeled with their inbound and outbound buffers. Cycle times for cranes are taken from tested statistical distributions at our industrial partner.
- *Workstations*: workstations are modeled with their inbound and outbound buffers.
- *Divert*: we model the first entry point of TSUs with an *arrivals source* to generate TSUs and a divert that makes decisions on routing TSUs to the sorter or to the ASRS.

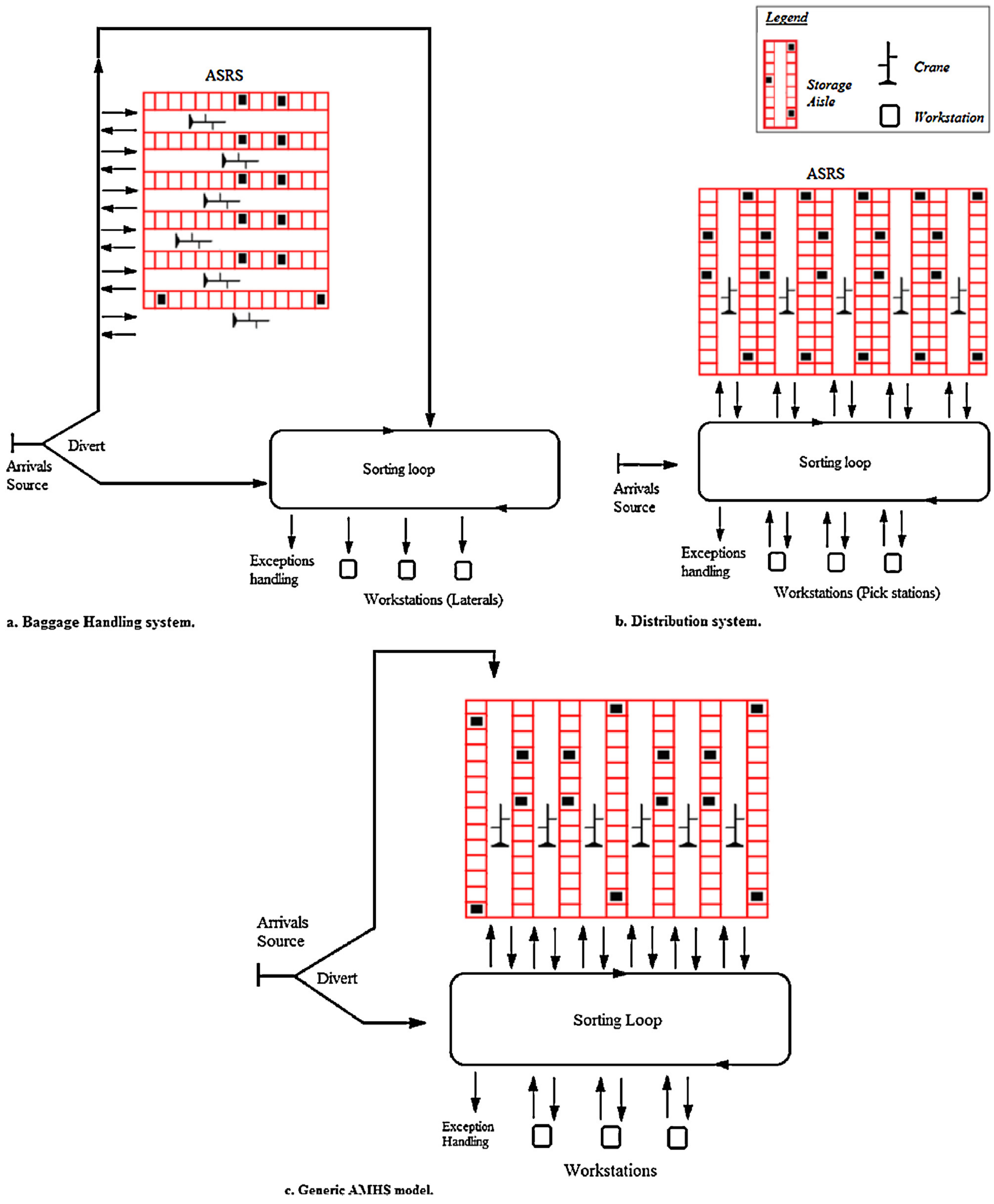


Fig. 5. AMHS models.

- *Sorting loop*: the loop is modeled at some level of aggregation, where we keep track of the loop capacity and travel times on the loop, but we do not keep track of the location of every point on the loop at every moment in time.
- *Exception handling outfeed*: this is a special type of outfeed conveyor where TSUs can be diverted in some cases, e.g., when a bag misses its flight.

3.3. Decision-making processes

In this section, we highlight the main decision making processes relevant to our model in Baggage Handling and Distribution. Although these are two different industrial sectors, we analyze how far we can model decision-making processes in a generic manner. We present the decision-making processes at each level of control.

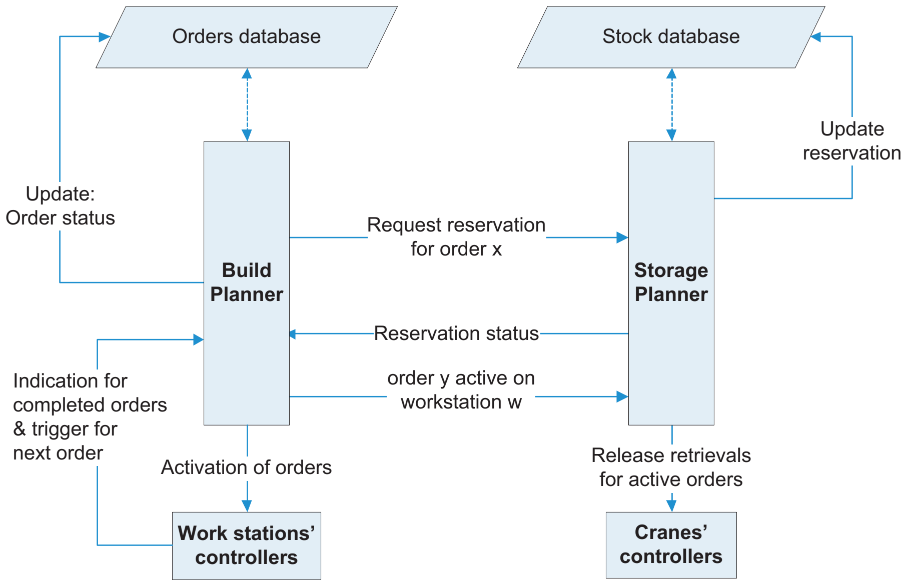

Orders database
Stock database
Update reservation
Update: Order status
Request reservation for order x
Build Planner
Storage Planner
Reservation status
Indication for completed orders & trigger for next order
order y active on workstation w
Activation of orders
Release retrievals for active orders
Work stations' controllers
Cranes' controllers

Fig. 6. Communications at the planning level.

3.3.1. *Planning processes*

At the Planning level (see Fig. 6), we identify two main processes: planning the outbound flow from the ASRS, and planning the inbound flow to the ASRS.

3.3.1.1. Planning the outbound flow from the ASRS. This process is a planning process as it requires a global view of the ASRS and of the destination workstation(s). Moreover, it results in assigning tasks to resources, e.g., retrieval tasks to cranes. There are two main subprocesses in outbound flow planning:

(a) *Stock reservation*: in Distribution, a customer order has a set of order lines, each referring to an SKU required with a certain quantity. An order is built on one workstation, but to build the order, stock is retrieved from the ASRS. Since multiple TSUs may hold the same SKU, it is necessary to decide on which TSU to *reserve* for usage of a certain order, i.e., stock reservation. However, in Baggage Handling we define an order as a set of *bags* required for a certain flight. In this sense, bags are uniquely identified, as each bag entering the system via check-in desks or as transfer baggage is already assigned to a specific *order* (flight). Therefore, we see the stock reservation as a process that results in bringing the distribution system to the same level of detail as a baggage handling system, by assigning TSUs to orders. This process is accomplished as the build planner requests stock reservation for certain orders (plans orders) from the storage planner, which in turn looks for TSUs to reserve. Typically, broken TSUs are attempted before breaking a full TSU, as many broken TSUs in the ASRS mean a loss in storage capacity. The build planner makes sure that a couple of orders are planned and ready for activation on any workstation requesting work.

(b) *Order release*: workstations trigger the build planner to activate orders, based on work progress in Distribution, but according to time schedules in Baggage Handling. As soon as an order is active on a workstation, stock belonging to this order has to be released from the ASRS. Therefore, the build planner informs the storage planner that a certain order is active. In turn, the storage planner dynamically assigns the reserved TSUs to candidate cranes as retrieval tasks, i.e., if the reserved TSU is accessible by more than one active crane, the storage planner assigns the retrieval to the crane having less workload. From this point on, cranes are responsible for executing and sequencing these tasks in the scheduling level of control.

3.3.1.2. Planning the inbound flow to the ASRS. When a TSU requires storage, it is announced to the storage planner, which responds with a destination aisle and crane to perform the storage operation. The decision can be made according to different control rules. At the planning level, decisions are based on detailed information, i.e., SKU levels in each aisle, and on comparisons between different possible aisles to store in. However, it is possible to perform this process at the scheduling level using simple rules, e.g., round-robin. The advantage is then simpler software, but we have to test if such simple rules do not cause any deterioration in the system performance. As a matter of fact, the outbound flow is the main contributor to system throughput, and assigning the inbound flow of TSUs to aisles may not have a *direct* impact on the outbound flow. Section 4 presents and analyzes alternative control rules further.

3.3.2. *Scheduling processes*

At the scheduling level (see Fig. 7), we identify two main processes: scheduling crane retrievals, and routing arrivals.

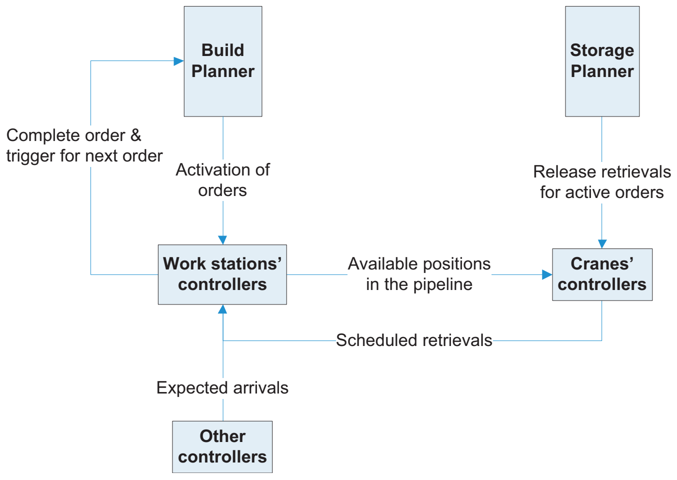

Build Planner
Storage Planner
Complete order & trigger for next order
Activation of orders
Release retrievals for active orders
Work stations' controllers
Available positions in the pipeline
Cranes' controllers
Scheduled retrievals
Expected arrivals
Other controllers

Fig. 7. Communications at the scheduling level.

3.3.2.1. Scheduling crane retrievals. Given a set of retrieval tasks, crane controllers schedule these tasks based on their priority and the pipelines of destination workstations. The criteria for priority differ according to the industry and are explained further in Section 4. The pipeline size of a workstation is the number of TSUs that are in transport to this destination at any point in time: we only send more TSUs if the number of TSUs already in the pipeline is less than the pipeline size, in order to prevent overloads and congestion. We use the pipeline size concept often in our control in the following manner: each workstation receives information about incoming TSUs (from cranes or the receiving divert) and in turn updates the number of empty positions remaining in its pipeline. This also includes scheduled retrievals that are not physically in the pipeline yet. Other controllers, e.g., crane controllers, observe the pipeline capacities and take this information into consideration when scheduling crane operations. The size of the pipeline is an important parameter to define in the control architecture. In Distribution, the sorter system is small and flow has to be strictly controlled. Therefore, the pipeline size is typically equal to the *number of locations in the inbound buffer of the workstation*. In this way, if any problem occurs in the workstation or the operator is temporarily absent for some reason, then all TSUs in transport can be accommodated in the inbound buffer. No TSU should waste loop capacity by circulating on it due to blocked entry to the workstation. On the other hand, in Baggage Handling, there are long transport routes from the ASRS to the workstations area, and the sorter loop is much larger, and shows fluctuating occupation levels according to baggage arrivals and flight schedules. In such a system, it is important to maintain a continuous flow toward workstations during the make-up time. Otherwise, there will be instances of no flow causing delays in baggage delivery. Another point is that travel times to workstations may differ due to the larger system. In this sense, the farther workstation needs more TSUs in transport than the nearer one to maintain a continuous and balanced flow. Therefore, for Baggage Handling, we define the pipeline size for workstation i as the capacity of the workstations in *ipm* (items per minute) times the average transport time to destination in minutes (average over all cranes and the arrivals divert, and without actual traffic delay considerations):

$$PipelineSize_i = CapacityWS_i \times AverageTransportTimeWS_i \tag{1}$$

When the pipeline to a certain destination is full, system controllers react by blocking further TSUs from being retrieved or routed to this destination. Therefore, the pipeline size is an important parameter that is used to control material flow in the system, and to prevent overflows.

3.3.2.2. Routing arrivals. In Baggage Handling, as mentioned earlier, arrivals are routed either to the sorter system or to the ASRS. This choice is made by the arrivals' divert controller, using system information and status of destinations. We route arrivals directly to the sorter system if the make-up for the corresponding flight is open and the pipeline(s) of the destined workstation(s) is (are) not full. If one of these two conditions is not satisfied, then we route to the ASRS, and delegate the scheduling task to crane controllers there. If more than one workstation is available, then we route to the one with the least occupied pipeline.

3.3.3. Local traffic control

In this material flow control model, local traffic control deals with two main processes:

- *Space allocation on the loop*: the loop controller has to allocate free spaces on the loop to TSUs waiting to enter the loop from outbound buffers of cranes and workstations.
- *Crane storage cycles*: the crane executes storage cycles to store TSUs waiting on its inbound buffer. Storage does not need to communicate with other system components for information, and can execute this process locally given TSUs to store and available storage locations in the designated aisle.

We note that schedulers are the controllers responsible for workload control, because they decide on task execution times, e.g., retrievals. Pipeline size limitations reflect a pull system for material flow. Traffic controllers have to deal with material physically moving as a result of scheduling decisions, and do not influence the amount of material in transport.

4. Implementation

4.1. Experimental setup

We use the UGS-Tecnomatix Plant Simulation software to build a simulation model, which is scalable to different system sizes, in terms of the number of aisles, cranes, and workstations. Moreover, our model is flexible to different system settings regarding layout configurations (e.g., accessibility of cranes to storage racks) and capacities of system's resources (e.g., buffers and cranes). In order to model a certain system, the following modeling parameters have to be configured in our simulation model:

- Inbound/outbound buffer sizes of workstations and cranes.
- Aisle capacities, in number of storage locations per aisle.
- Crane capacity, in number of TSU locations the crane has. In our model, a crane can carry 2 and 4 TSUs simultaneously in Baggage Handling and Distribution respectively.
- Workstation capacity in terms of the number of TSUs processed per minute.
- Sorter loop speed, and size in number of TSU locations available.
- Travel times matrix: this matrix provides travel times on the sorter loop for every source and destination pair.
- Aisle-crane accessibility matrix, this matrix shows what cranes are operational in what aisles. In Distribution, an aisle is accessible by one crane only. However, in Baggage Handling, due to high reliability requirements, a redundant system design where two cranes can access the same aisle is in place.

Moreover, the following parameters are control parameters, which are implemented in the simulation model, as well as in the actual software of AMHSs:

- *Maximum number of orders simultaneously active on a workstation*. In Distribution, it is common to have multiple orders processed simultaneously. In our distribution system setting, 6 orders are active simultaneously on a workstation. In Section 4.3, we discuss an implication of this parameter.
- *Planned orders threshold*. This refers to the number of orders ready for activation on any workstation requiring work. These are to be planned in advance by the build planner (in Distribution), and are typically equal to the number of active workstations. This is the minimum number of orders needed for activation on any workstation requiring a new order. We keep the minimum in order to dynamically plan orders based on the status of the system.

For Baggage Handling, we configure the settings to model a baggage handling scenario according to a major European airport, which was a reference system for our study. This system has 18 workstations and 13 cranes operating on 12 storage racks, so that each storage rack is accessible by two cranes. We use data sets regarding flight schedules and baggage arrivals from the same system, for a whole day operation. An order in Baggage Handling refers to the baggage required for a certain flight, where each order line is a unique bag that is assigned to the flight.

For Distribution, we configure the settings of our model to model an existing automated distribution center in the Netherlands, which has 3 workstations, and 5 cranes in 5 storage aisles. For data sets in Distribution, we run experiments according to one of the following order structure scenarios, for one day shift:

- Big orders: 15–25 order lines/order.
- Small orders: 1–5 order lines/order.

We assume that to satisfy an order line, one TSU (holding the required SKU) is always sufficient. This assumption is based on common practice and modeling cases from our industrial partner. We are interested in material flow and the logistic control of the system, so issues related to inventory profiles, i.e., the number of different SKUs, are not an interesting factor to vary in our model. We use 1000 different SKUs in all experiments.

Experimental control rules for assigning inbound TSUs to aisles are as follows:

1. *SKUs distribution*: this rule means selecting the aisle containing the fewest TSUs of the incoming SKU and it is current practice in Distribution. For ties, the criterion becomes the total number of TSUs in the aisle. In Baggage Handling, this level of detail, e.g., to assign bags to an aisle based on bags of the same flight, is not sensible (for practical reasons) and not applied in practice.
2. *Aggregate totes distribution*: In Distribution, this means to select the aisle having the least number of *broken* TSUs in aggregate terms regardless of the SKU content (ties are broken as in rule 1). In Baggage Handling, this rule means simply selecting the aisle having the smallest number of TSUs in aggregate terms.
3. *Round-robin*: simply store in aisles sequentially for every incoming TSU regardless of the number of broken TSUs or SKU distribution.

In Distribution, using the aggregate number of *broken* rather than *full* TSUs in the aisle makes more sense (control rule 2), because stock reservation always looks for broken TSUs first, and then distributing these over aisles contributes to workload balancing among cranes. In all control rules, there should be storage locations available in the selected aisle, and at least one crane active on the aisle to perform the storage operation. Otherwise, the aisle is not a candidate.

4.2. General results on performance

4.2.1. Distribution

We observe that there are no significant differences in terms of throughput, when considering various control rules for inbound flows. Fig. 8 shows the number of order lines processed in every hour of operation (each column represents throughput of an hour), for each workstation, order size, and inbound control rule. Naturally, the last hour has fewer order lines processed due to the end of a shift, and is excluded from performance measures. The average number of order lines processed per hour was 314 and 320, with 87% and 88% utilization of workstations, for small and big orders respectively.

We note that smaller orders cause throughput levels to decline. To analyze this trend further, consider orders that are extremely small (1 order line per order), then 6 orders active on a workstation lead to only 6 TSUs in transport. In this case, as an order is picked and a new order is to be activated, more delays are expected until the next TSU is scheduled, retrieved, and transported. When the order is big, each order requires several TSUs that are in transport and keep the workstation busy. We experimented with orders of 1 order line each, and found that the number of picks per hour drops to 271, and utilization of workstations drops to 75%.

Moreover, we notice that in the small orders scenario the number of picks tends to decrease as time goes by (see Fig. 8). This is explained because, when orders are small, the number of orders that have to be processed is higher, in order to generate a similar number of total order lines to simulate. Therefore, the probability that several orders simultaneously need the same SKU becomes higher. In this case, as more TSUs are simultaneously needed by several orders, delays may occur until reserved TSUs are available for a specific order. Note that there is a limited number of TSUs per SKU in the system.

With regard to inbound control rules, rules 2 and 3 do not show any deterioration in throughput compared to rule 1 which is common practice. Hence, we may recommend using one of these rules as they require simpler software implementation and result in synergy in control with Baggage Handling. At this point, we recommend abandoning rule 1, but recommending rule 2 or 3 depends on the results from the baggage handling scenario.

4.2.2. Baggage handling

We found that the pipeline size proposed in Section 3.3 is not large enough, causing insufficient material flow in the system. Many bags were missed, and the utilization of system resources was low. The explanation is that workstations were often idle, because TSUs did not keep flowing according to the theoretical equation. The equation assumes a free flow situation, where in reality delays occur, e.g. due to waiting times to merge on the loop which in turn may cause blocking entrance of new bags due to the maximally allowed pipeline size. Moreover, TSUs that are scheduled and not physically in transport contribute to pipeline occupation.

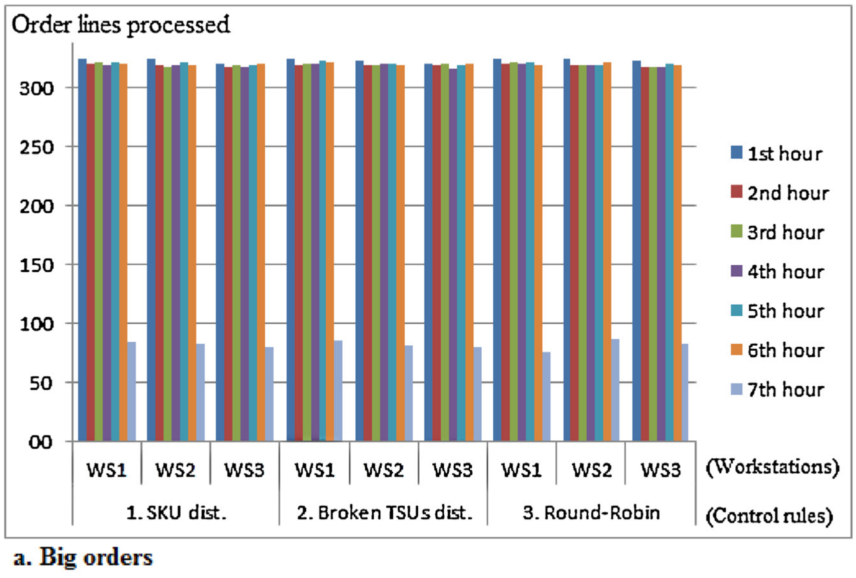
Therefore, we introduce a time allowance to be added to the average transport time in Eq. (1). The extra time allowance accounts for retrieval time of cranes and traffic delays (Eq. (2)). Based on simulation results, we find that a time allowance of 2 min leads to a good performance, with an irregularity rate of 0.095 bags lost in every 1000 bags, in both rules 2 and 3. However, this time allowance is a layout-specific configurable parameter.

$$PipelineSize_i = CapacityWS_i \times (AverageTransportTimeWS_i + TimeAllowance) \qquad (2)$$

Inbound control rule 2 or 3 does not have significant effects on the performance of the baggage handling system.

4.3. Genericity analysis

We have implemented the proposed control architecture in the material flow model, to analyze to what extent a generic software can be maintained for the two different industrial sectors: Baggage Handling (BH) and Distribution (D). We focus on the analysis of our software code that deals with the logistic planning and control of the system which is often different per industrial sector, i.e., decision-making functions, excluding the code we implemented for other purposes, e.g., system initialization, creating data sets. Moreover, we do not analyze Application Programming Interfaces (APIs) or Graphical User Interfaces (GUIs) that need to be provided toward the hardware (see Section 2.1). The latter applications are standard interfaces that are available at our industrial partner and at other software developers. These applications are used to drive the hardware, but are not part of the decision-making functions. However, we provide standard interfaces between the



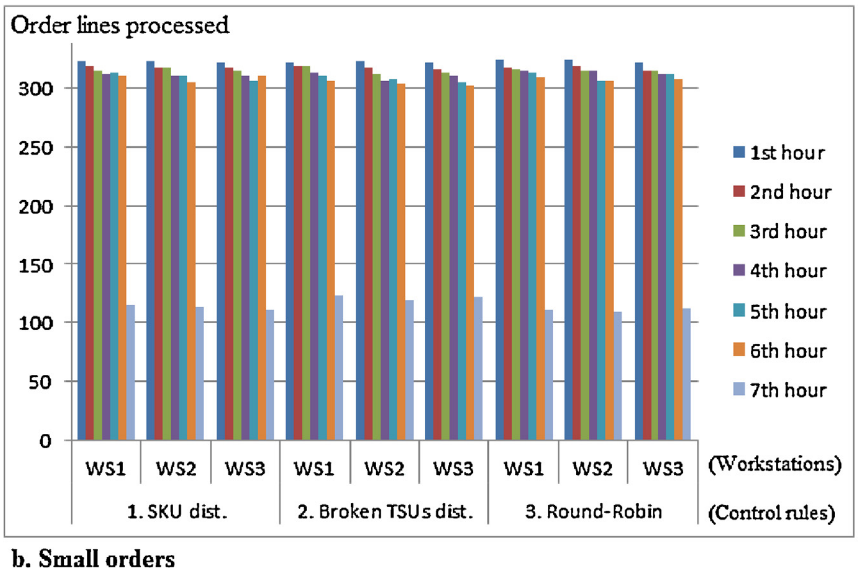


Fig. 8. Throughput levels per workstation under different control rules.

decision-making procedures, and decision-making units (controllers). In other words, we focus on the decision-making functions in detail (the control architecture), but not on the implementation of the control architecture in a real-life software and connecting it to a real-world installation (the software architecture).

Our perspective on software is therefore on an abstract level and deals with the code we used to implement these decision-making functions. This perspective supports our generic approach to decision-making in providing a generic control architecture, where connecting this control architecture to the hardware of a certain AMHS is a different issue.

Our target is to keep the control generic, but at some point we had to deviate to satisfy sector-specific requirements. As a matter of fact, for decision-making processes that exist in both industrial sectors, we managed to implement a software code of which 84% (in terms of lines of code) is used identically by both sectors, while the remaining 16% vary. We present these percentages to give insight into the potential synergy between the different industrial sectors. These percentages are dependent on the implementation in our simulation, and so they may vary in other implementations or other simulation models or packages. However, the generic design of our decision-making processes leads to a high degree of synergy even with different implementations.

We note that our analysis is focused on the software related to decision-making functions, which addresses the differences in these sectors. As we claimed earlier in Section 2.1, standardization in software that is not related to decision-making should be the rule rather than the exception, and so including it in our analysis has to bring even more synergy.

We claim that our control architecture is generic in its applicability to different systems, in the sense that implementers need to understand and customize only 16% of the code, where sector-specific elements need to be handled. Implementers should understand the majority of the code (roughly 84%) at a high level using standard procedures and reusable modules. To explain this claim, we use Fig. 9 in which we provide a representation of the relevant decision-making procedures, where we include the main decision-making procedures for the main controllers we have in the model (the build planner, the storage planner, an object from the crane controller class, an object from the workstation controller class, the loop controller, and the arrivals divert controller). To keep Fig. 9 readable, we do not show many of

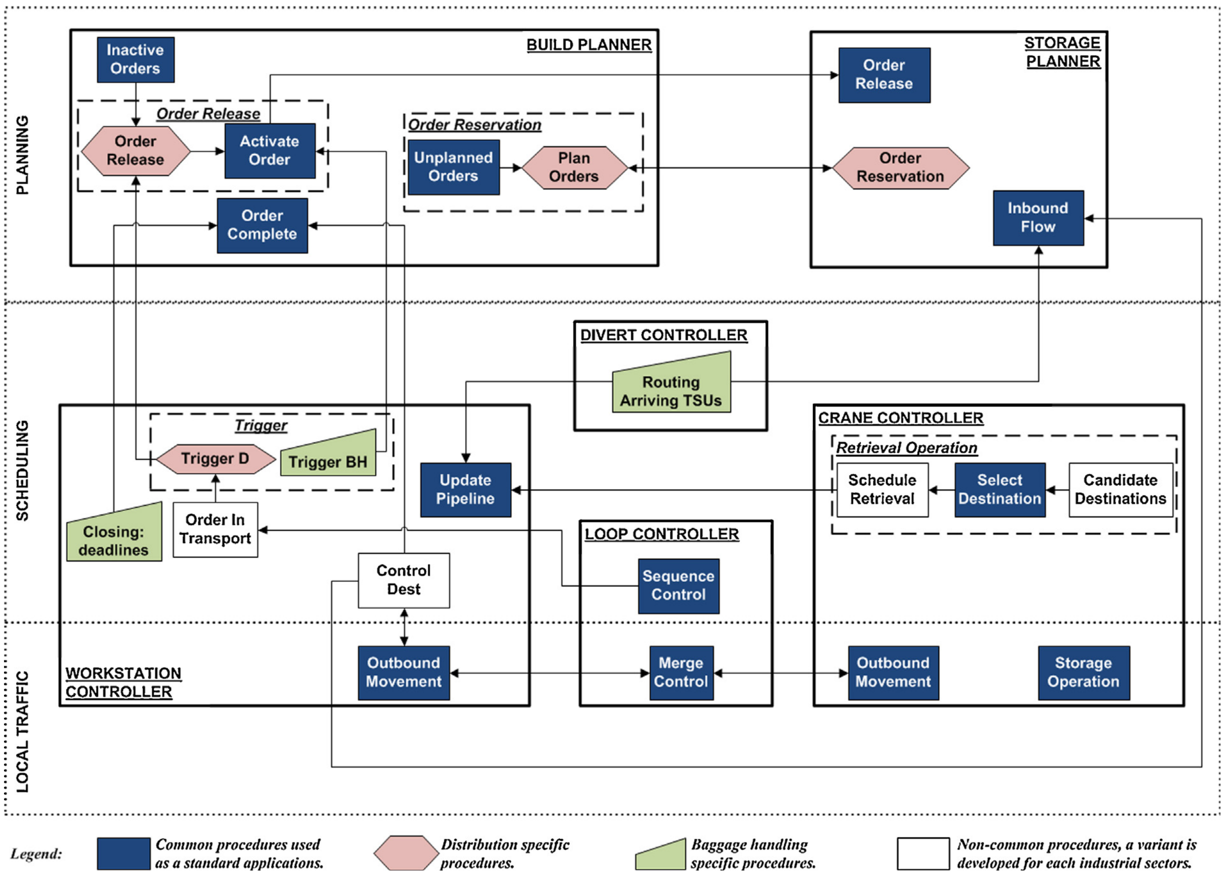


Fig. 9. Main decision-making procedures in the control architecture.

the standard procedures, databases and their connections, or the communication links on assigning orders to workstations and retrieval tasks to cranes (shown in Fig. 6).

At this point, we need to further analyze decision-making procedures where the software is not the same, and explain why is it inevitable to deviate from the generic code. Therefore, the remainder of this section is dedicated to analyze the decision-making procedures that are not standardized for both industrial sectors (16% of the software code). Moreover, we analyze the procedures that can be incorporated or omitted depending on the need for them in the form of a plug-and-work mechanism.

4.3.1. *The order reservation process*

At the planning level, Fig. 9 shows the ‘order reservation’ process as a non-common process in both the build planner and the storage planner. As mentioned earlier, this process brings Distribution at the same level of detail as Baggage Handling.

Fig. 9 shows the components of this process in the build planner, i.e., the ‘Unplanned Orders’ and the ‘Plan Orders’ procedures. The ‘Unplanned Orders’ procedure checks (in connection with the orders database) whether there are still unplanned orders, and triggers the ‘Plan Orders’ procedure to plan new orders given the threshold conditions (see Section 4.1). This procedure is active in our model for the two industrial sectors. However, in Baggage Handling it always finds that all orders in the database (in this case flights) are planned, and so it never triggers the ‘Plan Orders’ procedure and in turn the ‘Order Reservation’ procedure in the storage planner is never used. These procedures are plug-and-work like procedures than can be simply removed from the architecture when implemented in Baggage Handling.

4.3.2. *Triggering and releasing orders*

In this section, we discuss two elements of Fig. 9 together, because they are closely connected. First of all, the trigger process in the workstation controller differs per industrial sector, because this process is directly related to the different operational environment, which we described in Section 1.2. This is summarized as follows:

- In Distribution: the trigger to activate a new order, and the announcement of a complete order is based on work execution (late TSUs are waited for).
- In Baggage Handling: orders are triggered to start and are declared completed based on time schedules (late TSUs are missed).

Therefore, in Fig. 9 there are two variants of the ‘Trigger’ procedures, one per sector. This is an unavoidable sector-specific application. In Distribution, the ‘Sequence Control’ procedure in the loop controller sends update messages about TSUs in transport to the destination workstation. Specifically, the ‘Order In Transport’ procedure receives these messages. In Distribution, the ‘Order In Transport’ procedure checks whether all of the TSUs of an order are in transport and sends a message to the ‘Trigger D’ procedure. In turn, the ‘Trigger D’ procedure checks whether a new order should be activated and, if so, sends a message to the higher level build planner asking for a new order. This message is received by the ‘Order Release’ procedure. The latter procedure checks whether there are still any planned but inactivated orders. If so, then it selects an order to activate and sends a message to the ‘Activate Order’ procedure to activate the selected order on the triggering workstation.

In Baggage Handling, the ‘Order In Transport’ procedure does not send any outgoing messages, and therefore part of the code in this procedure is unused for this system model. As a matter of fact, in Baggage Handling the orders (in this case flights) are planned beforehand according to certain time schedules. Therefore, at certain moments in time the 'Trigger BH' procedure sends a message to the build planner to activate an order (in this case the planned flight). Since it is already known what order is to be activated, this message is received directly by the 'Activate Order' procedure in the build planner.

In this section, we observe that in different industrial sectors we use the relevant variant of the 'Trigger' procedure in the workstation controller. Moreover, we find that the 'Order Release' procedure in the build planner is used only in Distribution to search for an order to activate, if any, on the triggering workstation. The 'Activate Order' procedure is standard and is used in both sectors.

4.3.3. Controlling TSUs that leave the workstation

As TSUs are processed at workstations, they have different options to proceed for each of the industrial sectors. The procedure 'Control Dest' (Fig. 9) is responsible for guiding these processed TSUs. In Baggage Handling, TSUs always leave the system, and so the 'Control Dest' procedure instructs low level controllers to move the bag. However, in Distribution processed TSUs are often broken TSUs, which need to be returned to the ASRS. Therefore, they need to be announced and re-routed via the 'Inbound' procedure. Moreover, if a TSU is the last TSU of a certain order, then 'Control Dest' sends a message that the order is complete to the build planner.

The 'Control Dest' procedure needs to have a variant per industrial sector. It reflects an unavoidable system-specific application due to the different operational environments.

4.3.4. Order complete in Baggage Handling

In Section 4.3.3, we showed how the order is announced complete in a distribution system. In Baggage Handling, the procedure 'Closing' (Fig. 9) is responsible for sending the order complete messages at certain moments in time, based on planned end times for flight loading operations.

4.3.5. Guiding arrivals

A sector-specific procedure 'Guide Arrivals' is added in Baggage Handling to route arriving bags (see Section 3.3), because in Baggage Handling a scheduling decision has to be made on routing bags either directly to workstations or to the early bag storage.

4.3.6. Scheduling crane retrievals

Scheduling crane retrievals is a vital decision-making process that has to adapt to sector-specific requirements, although it has the same basic structure in the crane controller. To understand this process better, we highlight an important operational difference in the systems we studied: in Distribution multiple orders are simultaneously active on a single workstation, whereas in Baggage Handling a single order is active on one or more workstations simultaneously.

Fig. 9 shows that the retrieval operation consists of three main procedures, executed in each retrieval decision:

1. *Defining candidate destinations* (*for which a retrieval can be scheduled*): in Distribution, the destinations of all active order line retrievals define the candidate destinations. In Baggage Handling, we have to serve the most urgent flight first, and so those workstations on which the urgent flight is active are the candidate destinations.
2. *Selecting a candidate destination*: the selection among candidate destinations is identical for both industrial sectors and depends on least occupied pipeline.
3. *Scheduling a retrieval*: in Distribution, we schedule a retrieval to the selected destination, based on the oldest active order first. In Baggage Handling, we schedule a bag for the selected workstation and the selected (urgent) flight.

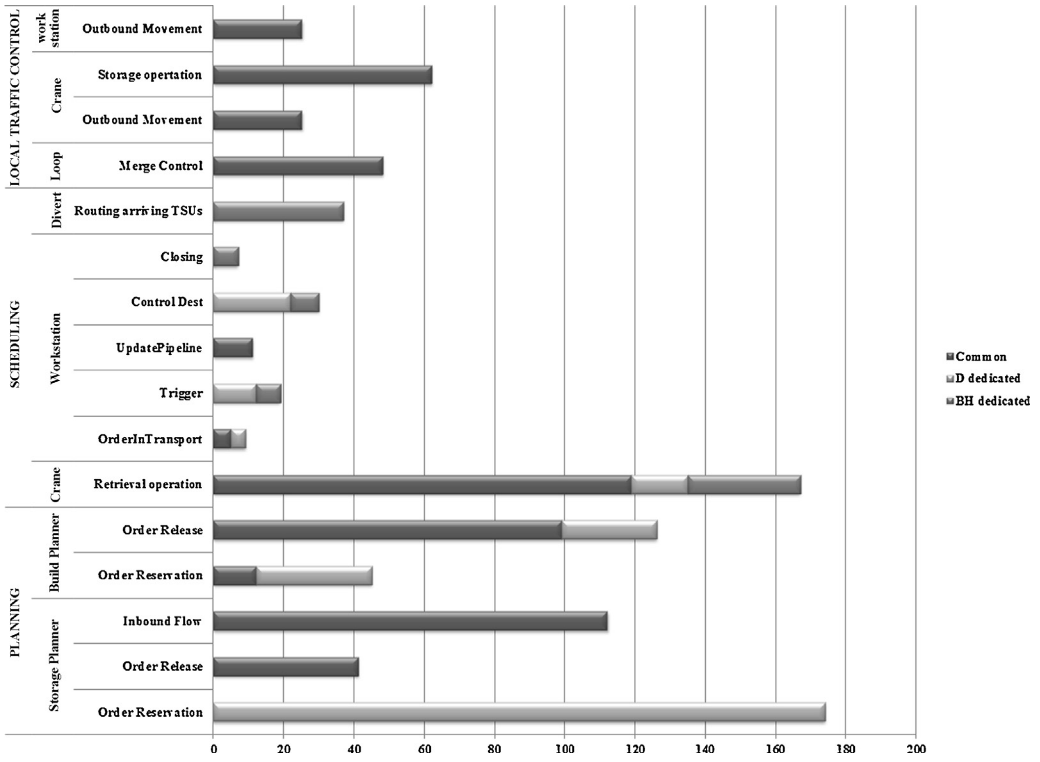


Fig. 10. Commonality of code.

In this process, the second procedure is standard and applied in both industrial sectors. However, for each of the first and third procedures above we have a variant for Baggage Handling and a variant for Distribution.

Having discussed the decision-making procedures in detail, we find it useful to come back to the decision-making processes in generic terms (see Section 3.3), and provide an overview of the commonality in software for these processes according to the implementation in the simulation model. Fig. 10 presents the software coverage, i.e., commonality in the lines of code. In this sense, Fig. 10 presents the number of lines of code in common, and the numbers of lines dedicated per sector, for every main decision-making process. We note that decision-making processes at the local traffic control are common for both sectors, which should be the case for such lower level decisions that deal mainly with the movement of TSUs on the equipment. Variations occur at the planning and scheduling levels, where the processes that are divided between the Baggage Handling or the Distribution process with no common segments indicate that these processes require a variant per sector as described in the aforementioned analysis. On the other hand, processes that have dedicated segments per sector but also common segments mean that certain procedures within these processes are common while others require a variant per sector. Finally, processes that are entirely dedicated refer to plug-and-work procedures.

5. Conclusion

In this paper, our main target was to provide a proof-of-concept for the applicability of generic control for AMHSs in different industrial sectors. To this end, we presented a material flow model that is applicable in two different industrial sectors, and applied generic control procedure on it building upon an earlier study, which proposes a generic control architecture.

We first provided a description of our overall project, and a description of the processes in the industrial sectors we address (Section 1). Thereafter, we defined our theoretical framework and presented a literature review of relevant studies (Section 2). In Section 3, we presented the control architecture, the system model, and the decision-making processes. Afterwards, we explained the implementation of our control and general results (Section 4), and analyzed the genericity of our control approach in detail.

As a matter of fact, we managed to achieve a high level of synergy in control over common decision-making processes. This is testified by a software code that is 84% identical for both the Baggage Handling and the Distribution sectors. The generic structure of our decision-making processes facilitates a minimal deviation from the generic code. This is achieved because differences in decision-making criteria and control rules among industries are easily integrated in specific procedures that do not hamper the overall structure of the decision-making process. However, these differences are due to sector-specific parameters, e.g., plane departure schedules, which we have to adapt to.

In addition, we found that inbound flow control in Distribution can be implemented in synergy with Baggage Handling. Inbound flow control can be implemented at the planning level using aggregate information about TSUs in storage, or at the scheduling level using a simple scheduling rule, i.e., round-robin. Uncommon decision-making processes that are specific to one sector, e.g., order reservation, are inevitable differences. These processes are built as functional add-ons to the generic control architecture, which do not hamper the generic structure and do not need special interfaces as we have standardized communications between controllers.

We note that our data structures should be flexible enough to adapt to unforeseen future applications. Therefore, in future research we plan several extensions on the basic model of this paper in order to test our control architecture and its genericity. In a future study, we extend the current model to include a routing model in a complex transport network upstream the storage and sorter systems. In this extension, we model scheduling controllers of transport resources further. Moreover, we plan to make a comprehensive application of the generic control architecture to a BHS, where we introduce robots as a new type of workstations that are more in synergy with workstations in Distribution.

References

[1] Y. Alsafi, V. Vyatkin, Ontology-based reconfiguration agent for intelligent mechatronic systems in flexible manufacturing, Robotics and Computer-Integrated Manufacturing 26 (4) (2010) 381–391.

[2] F. Amato, F. Basile, C. Carbone, P. Chiacchio, An approach to control automated warehouse systems, Control Engineering Practice 13 (10) (2005) 1223–1241.

[3] R.N. Anthony, Planning and Control Systems: A Framework for Analysis, Harvard Business School Division of Research, Boston, 1965.

[4] R.F. Babiceanu, F.F. Chen, Performance evaluation of agent-based material handling systems using simulation techniques, in: Proceedings of the Winter Simulation Conference, Orlando, FL, December 2005.

[5] R.F. Babiceanu, F.F. Chen, R.H. Sturges, Framework for the control of automated material-handling systems using the holonic manufacturing approach, International Journal of Production Research 42 (17) (2004) 3551–3564.

[6] X.W. Chen, S.Y. Nof, Prognostics and diagnostics of conflicts and errors over e-Work networks, in: Proceedings of the 19th International Conference on Production Research (ICPR-19), Valparaiso, Chile, July 2007.

[7] R. Claes, T. Holvoet, D. Weyns, A decentralized approach for anticipatory vehicle routing using delegate multi-agent systems, IEEE Transactions on Intelligent Transportation Systems 12 (2) (2011) 364–373.

[8] D.M. Dilts, N.P. Boyd, H.H. Whorms, The evolution of control architectures for automated manufacturing systems, Journal of Manufacturing Systems 10 (1) (1991) 79–93.

[9] K. Furmans, F. Schönung, K.R. Gue, Plug-and-Work material handling systems, in: K.P. Ellis, K. Gue, R. de Koster, R. Meller, B. Montreuil, M. Ogle (Eds.), Progress in Material Handling Research: Proceedings of 2010 International Material Handling Research Colloquium, 2010, pp. 132–142.

[10] A. Giret, V. Botti, Holons and agents, Journal of Intelligent Manufacturing 15 (5) (2004) 645–659.

[11] J. Gu, M. Goetschalckx, L.F. McGinnis, Research on warehouse design and performance evaluation: a comprehensive review, European Journal of Operational Research 203 (3) (2010) 539–549.

[12] K. Hadeli, P. Valckenaers, M. Kollingbaum, H. Van Brussel, Multi-agent coordination and control using stigmergy, Computers in Industry 53 (1) (2004) 75–96.

[13] K. Hallenborg, Y. Demazeau, Dynamical control in large-scale material handling systems through agent technology, in: IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'06), 2004, 637–645.

[14] S.W.A. Haneyah, J.M.J. Schutten, P.C. Schuur, W.H.M. Zijm, Generic planning and control of automated material handling systems, Computers in Industry 64 (3) (2013) 177–190.

[15] A.C. Hax, H.C. Meal, Hierarchical integration of production planning and scheduling, in: M. Geisler (Ed.), TIMS Studies in the Management Sciences: Logistics, North Holland–American Elsevier, Amsterdam, 1975, pp. 53–69.

[16] T. Hunter, Simulation model evolution a strategic tool for model planning, in: Proceeding of the Winter Simulation Conference, Lake Buena Vista, FL, December 1994.

[17] M. Jahangirian, T. Eldabi, A. Naseer, L.K. Stergioulas, T. Young, Simulation in manufacturing and business: a review, European Journal of Operational Research 203 (1) (2010) 1–13.

[18] M. Johnstone, D. Creighton, S. Nahavandi, Status-based routing in baggage handling systems: searching verses learning, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 40 (2) (2010) 189–200.

[19] B.I. Kim, S. Heragu, R.J. Graves, A.St. Onge, A hybrid scheduling and control system architecture for warehouse management, IEEE Transactions on Robotics and Automation 19 (6) (2003) 991–1001.

[20] H.Y.K. Lau, S.O. Woo, An agent-based dynamic routing strategy for automated material handling systems, International Journal of Computer Integrated Manufacturing 21 (3) (2008) 269–288.

[21] T.S. Meinert, G. Don Taylor, J.R. English, A modular simulation approach for automated material handling systems, Simulation Practice and Theory 7 (1) (1999) 15–30.

[22] D.Y. Mo, R.K. Cheung, A.W. Lee, G.K. Law, Flow diversion strategies for routing in integrated automatic shipment handling systems, IEEE Transactions on Automation Science and Engineering 6 (2) (2009) 377–384.

[23] H.V.D. Parunak, Generation and analysis of multiple futures with swarming agents (2010), in: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10), Toronto, Canada, (May 10–14, 2010), pp. 1549–1550.
[24] V.N. Rajan, S.Y. Nof, Cooperation requirements planning (CRP) for multiprocessors: optimal assignment and execution planning, Journal of Intelligent and Robotic Systems 15 (1996) 419–435.
[25] B. Rouwenhorst, B. Reuter, V. Stockrahm, G.J. van Houtum, R.J. Mantel, W.H.M. Zijm, Warehouse design and control: framework and literature review, European Journal of Operational Research 122 (3) (2000) 515–533.
[26] A. Tařau, B. De Schutter, H. Hellendoorn, Centralized versus decentralized route choice control in DCV-based baggage handling systems, in: Proceedings of the IEEE International Conference on Networking, Sensing and Control, Okayama, Japan, March 26–29, 2009.
[27] A.N. Tařau, B. De Schutter, H. Hellendoorn, Hierarchical route choice control for baggage handling systems, in: Proceedings of the 12th International IEEE Conference on Intelligent Transportation Systems, St. Louis, MO, USA, October 3–7, 2009.
[28] H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, P. Peeters, Reference architecture for holonic manufacturing systems: PROSA, Computers in Industry 37 (3) (1998) 255–274.
[29] J.P. Van den Berg, A literature survey on planning and control of warehousing systems, IIE Transactions 31 (8) (1999) 751–762.
[30] P. Vrba, V. Mařík, Simulation in agent-based control systems: MAST case study, International Journal of Manufacturing Technology and Management 8 (1) (2006) p.175–p.187.
[31] D. Weyns, T. Holvoet, A. Helleboogh, Anticipatory vehicle routing using delegate multi-agent systems, in: IEEE Intelligent Transportation Systems Conference (ITSC 2007), September 30–October 3, 2007, 87–93.
[32] W.H.M. Zijm, Towards intelligent manufacturing planning and control systems, OR Spectrum 22 (2000) 313–345.
[33] E. Zimran, Generic material handling system, in: Proceedings of Rensselaer's Second International Conference on Computer Integrated Manufacturing, 1990.

S.W.A. Haneyah is a PhD Candidate at the Department of Industrial Engineering and Business Information Systems (IEBIS) at the University of Twente, The Netherlands.

J.M.J. Schutten is an Associate Professor at the Department of Industrial Engineering and Business Information Systems (IEBIS) at the University of Twente, The Netherlands.

P.C. Schuur is an Associate Professor at the Department of Industrial Engineering and Business Information Systems (IEBIS) at the University of Twente, The Netherlands.

W.H.M. Zijm is a Professor at the Department of Industrial Engineering and Business Information Systems (IEBIS) at the University of Twente, The Netherlands. Moreover, Professor Zijm is the Scientific Director of the Dutch Institute for Advanced Logistics (DINALOG).