CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

A Simulation Modeling Framework of Multiple-Aisles Automated Storage and Retrieval Systems

Jean-Philippe Gagliardi
Angel Ruiz
Jacques Renaud

December 2010

CIRRELT-2010-57

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

A Simulation Modeling Framework of Multiple-Aisles Automated Storage and Retrieval Systems

Jean-Philippe Gagliardi, Angel Ruiz, Jacques Renaud*

Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), and Department of Operations and Decision Systems, 2325, de la Terrasse, Université Laval, Québec, Canada G1V 0A6

Abstract. In this paper, we focus on modeling approaches for unit-load AS/RS. We first propose an up-to-date chronological literature survey encompassing models that use a form of simulation. We also introduce one of the first Object-Oriented Simulation Model of unit-load AS/RS, developed with a purely generic mindset. In this model, we separate the physical design decisions from operational decisions, making it possible to represent many scenarios of interest. We also propose possible extensions to the original model, allowing the study of other types of AS/RS, which are in use today.

Keywords. Multiple aisles automated storage and retrieval systems, unit-load, end-of-aisle, discrete-event simulation, object-oriented simulation, warehousing performance.

Acknowledgements. This research was partially financed by grants [OPG 0293307 and OPG 0172633] from the Natural Sciences and Engineering Research Council of Canada (NSERC). This support is gratefully acknowledged.

---

* Corresponding author: Jacques.Renaud@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec,
Bibliothèque et Archives Canada, 2010

1. Introduction

It is a great challenge to match supply and demand in today's turbulent markets. To do so, it is still necessary today to handle great amount of stocks. *Automated Storage and Retrieval Systems* (AS/RS) are used in modern distribution centers (DC) as they provide fast, accurate and efficient material handling, 24 hours a day. In its most basic depiction, an AS/RS consists of storage racks where loads are stored and retrieved automatically. This large definition is representative of the wide variety of possible system implementations, but this paper focuses on a particular configuration of AS/RS known as the unit-load AS/RS. Unit-load AS/RS are fully automated systems where a single storage or retrieval transaction involves only one unit-load. As these systems are often seen in high-volume distribution centers where unit-loads are finished goods pallets, they can also exist in production environments for work-in-progress storage. Figure 1 below illustrates the unit-load AS/RS concept under study.

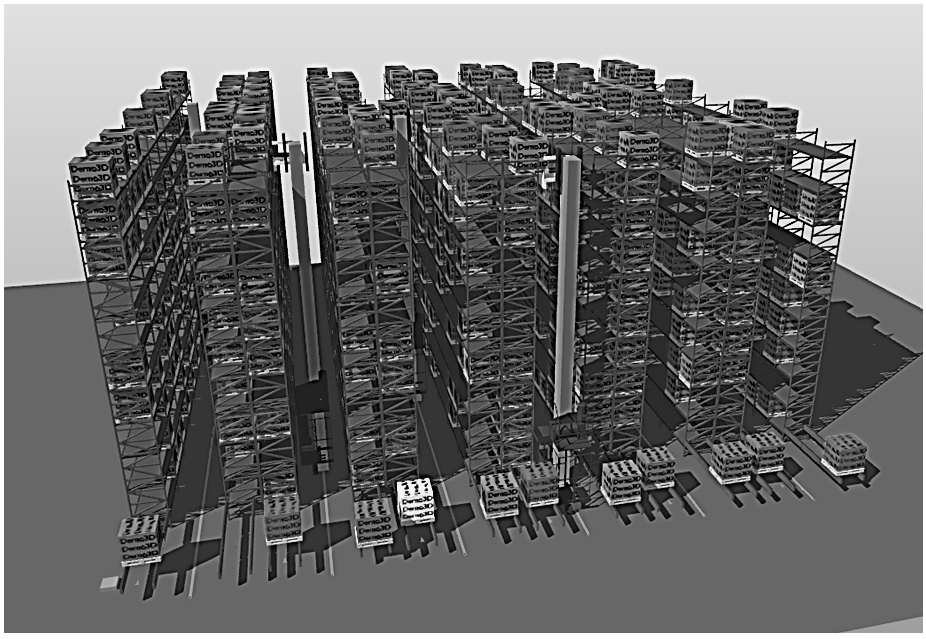

Figure 1: The unit-load AS/RS concept

When implementing a system like the one shown in Figure 1, many questions arise on both design and control levels. Many of these problems have already been tackled in the literature and a recent survey of the literature by Roodbergen and Vis (2009) wraps them up pretty well. From a design perspective, the configuration of the system is a great challenge. Designers must first configure the storage racks (number, length, height, depth, and spacing) which defines storage capacity and required floor space for the whole system. Also, the number of cranes and their degree of freedom

must be set. In some implementations, cranes may have the capacity to operate in several aisles, but in others cranes are aisle-captive. Another decision of crucial importance is the number and location of input and output (I/O) stations for cranes to interface with upstream and downstream links in the distribution supply chain. All these design decisions influence the implementation cost and will have a huge impact on the maximum throughput capability of the system. From an operational perspective, control decisions must be made in order for the system to achieve its maximum throughput capability for a given design. These problems include, among others: storage assignment, dwell-point positioning and requests sequencing. Storage assignment refers to the association of unit-load storage requests to empty storage locations. The dwell-point is the designated location for an idle crane to wait for the next request. Requests sequencing is the problem of dealing with the order in which storage and retrieval requests are treated by the cranes.

AS/RS design and control problems are difficult to deal with because AS/RS evolve in highly dynamic and turbulent environments where demand (translated in retrieval requests) is known with short notice. Like in other logistic processes, demand is also stochastic and is often subject to seasonality. Also, the state of the system changes within minutes. For instance, when sequencing requests at t_0, the set of available (empty) storage locations depends on the operations executed previously. The same phenomenon holds for retrieval locations where the set of locations that contain product *i* depends on the operations that were previously executed. These two factors tend to favor approaches aiming at formulating robust guidelines instead of solution methods seeking the optimal solution. All design and control decisions in AS/RS are linked to some extent; for instance, the number of cranes or their capability to move from one aisle to another greatly impact on the request sequencing process. These interactions make the joint evaluation of more than one decision very difficult. This fact opens a door to simulation studies on AS/RS where it is possible to compare many design and control combinations to assess their performance. This paper aims at proposing a detailed discrete-event simulation model for unit-load AS/RS and some of its possible extensions to cope with realistic settings. The model positions itself in complement to the current literature. As it can be observed in Gagliardi et al. (2010), in the papers where simulation has been employed, the main focus was put on the results and very little attention was given to the modeling approach. As a consequence, modeling approaches were not explicitly discussed. There is also a great number of critical assumptions that have not been mentioned and that can have a huge impact on experimental outputs of other scenarios.

The remainder of this paper is organized as follows. Section 2 surveys the literature on AS/RS simulation modeling. Section 3 presents an object-oriented simulation model in its general form.

Section 4 extends the general simulation model to study other configurations. Our conclusions are reported to section 5.

2. Literature review on AS/RS simulation

When modeling real-world applications, systems are often too complex to allow realistic models to be evaluated analytically. To overcome such a difficulty, strong simplifying assumptions are used in those models. For instance, in order to evaluate the performance of a given implementation, some analytical models assume a random storage policy and single aisle configuration which do not reflect most real-world implementations. Moreover, taking stochastic phenomenon into account (e.g. demand process) is sometimes an issue in pure analytical approaches. On the other hand, simulation is a numerical analysis method designed to evaluate the desired true characteristics of complex systems where some components behave stochastically. AS/RS conceptualization could certainly beneficiate from simulation analysis. This literature review focuses on modeling approaches that incorporate a form of simulation in the study of AS/R systems. As mentioned previously, the simulation models found in the literature are not detailed and give us the impression that they have been developed for very specific scenarios. The next paragraphs present some of these papers and highlight the lack of detailed models.

In past research, simulation has been successfully applied directly in AS/RS study. Among the first academics to employ the technique, Schwarz et al. (1978) were interested in examining and extending analytical models of Hausman et al. (1976) and Graves et al. (1977), by means of simulation on a discrete rack, in order to study storage assignment and interleaving effect on travel-time performance. Their analysis uses the same assumptions as their predecessors. They modeled a single aisle, then a two-sided aisle of 100 x 10 locations. Arriving unit-loads are placed in a storage queue of infinite capacity. At moment of storage of item *i*, the item length of stay (LOS) is generated. Using their model, the authors conclude that choosing an open location randomly or following the closest open location (COL) are equivalent at 90% rack utilization when neither interleaving nor sequencing is allowed. This type of conclusion is representative of the types of results found in the literature: a specific conclusion for a single configuration. In this case, it is legitimate to ask ourselves if these results are generalizable to other scenarios.

In some other cases, simulation has been employed in an optimization process of AS/RS. For instance, Azadivar (1984) proposes a simulation model in such a way that it is possible to both test the effect of assignment policies and optimize the system. The objective of the model is to optimize

the average round trip time required by the crane to complete a storage or a retrieval under a given distribution of length of stay of items on the storage racks. Stochastic approximations are used in a simulation optimization algorithm that obtains optimum variable values Using the continuous representation of a rack proposed in Hausman et al (1976), the simulation-optimization approach is run with 90% targeted rack utilization. The results confirm those of Schwarz et al (1978), since the same assumptions are used. This situation is also representative of the importance of assumptions in the field of AS/RS studies.

Randhawa et al. (1991) employed simulation to evaluate single and dual I/O point configurations in unit-load AS/RS. They study a unique single-sided aisle configuration. The authors propose 3 performance criteria: system throughput, mean waiting time and maximum waiting time. They assume that storage requests can only be processed with first-in first-out (FIFO) or COL rules. Retrieval requests can be arranged to minimize travel-time either working on a block composed of the first K requests (e.g. similar to the one presented by Han et al. (1987)), or by dynamically maintaining the list each time a new retrieval request arrives The simulation model is based on a discrete event engine. Studied configurations assume 75% rack utilization. Their results show that the shortest leg (SL) rule is better for all configurations studied. It is also shown that the dual-dock layout maximizes throughput. Another interesting element concluded from their simulation study is that, with respect to throughput, high rack utilization leads to reductions in throughput. The authors explain this result by the fact that, on average, the crane needs to travel longer to process requests. Nonetheless, results from all rack utilization levels were similar in nature. The conclusions are the same with respect to mean storage waiting time.

Knapp and Wang (1992) introduce petri-nets as a new technique to model AS/RS. It is implicitly assumed that the system under study is a unit-load AS/RS where a random storage assignment policy is applied within product classes. Single command cycles are also assumed, meaning that the crane always goes back to the I/O after each treated request. In their paper, Knapp and Wang (1992) explain Petri net principles extensively but give little conclusions on the AS/RS model. Later, Chincholar and Chetty (1996) also used stochastic colored petri nets in conjunction with the Taguchi method of design of experiments in order to study the effect of 5 control factors of the AS/RS on 3 performance metrics: time to complete M requests, unproductive travel-time and mean flow time. It is shown that these 3 metrics are mostly affected by the crane scheduling rules and crane operation mode.

Eben-Chaime and Pliskin (1997) question the fact that most studies on AS/RS focus on single aisle systems while giving little interest on other systems interfacing with the AS/RS. The authors extend on the integrative model proposed by Eben-Chaime and Pliskin (1996) composed of single depth racks, cranes, storage and retrieval queues and a list of empty slot addresses. The authors are mainly interested in evaluating 3 crane operating modes: single command (SC), dual command (DC) and hybrid command mode (HC), consisting of performing DC whenever possible and SC otherwise. It is explicitly formulated that the model can handle unit-load or mini-load operations. Results indicate that the HC mode is an efficient strategy to adopt in order to reduce storage requests queues length and keep good service levels.

Van den Berg and Gademann (2000) present one of the most complete simulation studies of control policies in AS/RS. They compare several storage assignment policies as well as the sequencing of storage and retrieval requests. The performance measure used throughout the study is the tradeoff between crane travel-time and response time of the system. It is assumed that the turnover of items is known and constant, and that the AS/RS is composed of a single two-sided aisle where the I/O point is located at a corner of the rack. The turnover rates, product mix and ordering policies are constant through time and the number of available unit-loads at period t is approximated by a Normal distribution (Van den Berg, 1996). It is also assumed that the system is balanced, i.e. the storage and retrieval requests queues are of equal length, and so dual command cycles can be formed. The model allows dynamic sequencing of the retrieval requests but in order to keep good service levels, urgency rules are employed (Han et al., 1987). The simulation model was implemented in C++ but no other details were given concerning its implementation except for the fact that there is at most a single unit-load of each item at all times. When a retrieval request is generated, a storage request for the same item will be generated at time t + TUNSi where TUNSi is a random variable with constant distribution. In the same manner, when a storage request is generated, a retrieval request will be generated at time t + TUNRi. Extensive simulation results are presented.

Meller and Mungwattana (2005) used simulation to evaluate the benefits of various dwell-point policies. The focus of their study is concentrated around the relation between the dwell-point policy and system utilization. The main performance measures employed are the mean time in system of retrieval requests and crane utilization as well. The authors assume that distances are measured in normalized units and if the crane receives a retrieval request while travelling to its dwell-point, it complete the trip before if processes the request. Dual command are formed with each request type being treated FIFO. The model is simulated for 100 000 time units in addition to 5 000 time units for warm-up. Results indicate that the dwell-point policy has insignificant $[2\%, 5\%]$ impact on system

response time when the AS/RS is highly utilized and also has insignificant (< 10 sec.) on the absolute response time for any utilization level.

Fukurani and Malmborg (2008) present an innovative method to estimate travel-times in a single aisle, unit-load AS/RS employing the queuing theory paradigm. They are interested in testing the 1-class Closest-open-location (COL) storage policy when rack utilization is less than 100%. In this case, the assumption of equiprobable access to any location used in PRS (Hausman et al., 1976) is not satisfied. In fact, the application of the COL policy tends to maximize utilization of storage locations closer to the I/O. It is assumed that, when system utilization is less than 100%, COL can yield shorter travel cycles than PRS. Their approach consists in modeling each storage location as a single server of a M/M/N Poisson queuing model, where N represents the number of locations of the aisle. The N locations are sorted in non-increasing order of their distance from the I/O point. The queuing model state distribution is then used to approximate the probability that individual locations are occupied. These probabilities are then applied in the estimation of storage and retrieval transaction travel-times under COL. In order to validate the approach, they compare the response of the proposed model to a discrete rack "simulation" model and to the APRS of Bozer (2010). Results indicate that the average response of the approach is closer to the discrete rack response than APRS. This paper is among the first to focus on modeling approaches for AS/RS studies.

Gagliardi et al. (2010) were interested in the storage assignment issue of unit-load AS/RS under the assumptions presented in section 2. To this end, the authors propose a detailed discrete-event simulation model that allows studying the performance of storage assignment policies under realistic conditions. Accent is put on the fact that the modeling approach is generic enough to test a wide variety of scenarios. The system is balanced, meaning that each retrieval request performed will trigger a subsequent storage request later in time. In order to validate the model, the authors first replicate the specific settings of Hausman et al (1976). The results are identical in nature as the latter, giving insight in the validity of the approach. These results suggest that a TBS storage assignment policy yields shorter crane travel-times than PRS. In addition to these results, the authors are interested in studying the situation where the amount of storage space per product is superior to 1 (LTPR>1). In this case, space allocation plays a bigger role in performance. By using different values of *s* (skewness parameter of the ABC distribution, e.g. Hausman et al. (1976), they show that the same conclusions do not always hold. In fact, it is showed that PRS can achieve better performance than TBS in some settings. These conclusions demonstrate the dangers associated with models that use assumptions that are not necessarily consistent with typical warehouses, as well as

showing that simulation models may give responses that are more precise than analytical travel-time models and are less restrictive concerning the necessary assumptions.

The contributions of this paper are twofold. First, we propose and discuss of a generic modeling approach of unit-load AS/RS using discrete-event simulation. The conceptual relations between physical aspects and decisional aspects are also discussed. Second, the model can be qualified as generic because the approach employs the object-oriented programming model and doing so gives us the possibility to study more than one scenario quite straightforwardly. To support this assertion, we present extensions that can be employed in that sense. The model proposed is especially useful in the case where the system consists of more than one aisle and expected crane travel-time is not a sufficient metric.

3. The object-oriented unit-load AS/RS simulation model

AS/RS are systems difficult to model analytically because they incorporate interactions of many subsystems, where more than one instance of some subsystems may exist. From this point of view, it is quite natural to adopt a system approach and modeling AS/RS as a collection of interacting objects. Another element of complexity is the presence of random events. As mentioned previously, simulation is a tool of choice when studying complex, dynamic and stochastic systems. Furthermore, since most events occurring in AS/RS are of discrete nature, discrete-event simulation seems like the best simulation modeling approach. In this section, we propose a discrete-event simulation model that employs the power of the object-orientation (OO) modeling paradigm (see Fujimoto, 2000) in order to study almost any system configuration with very light or no modifications. This section will mainly focus on unit-load AS/RS while section 4 will discuss possible extensions to the model. The model could be used “as is” in a simulation study or could be paired to a decision model in a simulation-optimization process (see Rosenblatt et al., 1993). The model also could be implemented in any object-oriented language (e.g. VB, C#...) or in any commercial simulation software which integrate the OO concept (e.g. Simio, AnyLogic, etc.). The model is articulated around physical parts, decisional parts and inbound (production) and outbound (demand) interfaces.

Figure 2 gives a macro view of the AS/RS object-oriented model. The model is a compound of a crane controller, a demand model which generates retrieval requests and a supply model responsible for the generation of storage requests. A request is dispatched to a specific instance of a crane following the set of rules implemented in the crane controller. The cranes are responsible for the execution of the requests which, in turn, modify the state of the aisles by changing the content of the

storage locations. In the Figure, informational transactions are represented by dotted edges, while continuous edges represent physical unit-load transactions. In the next paragraphs, we will discuss the nature of each component in further details.

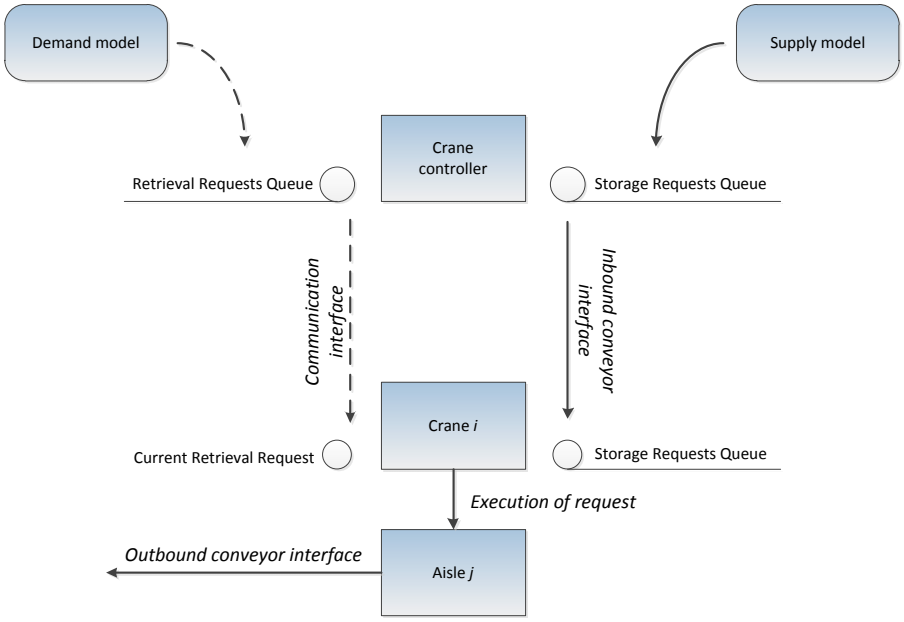

Figure 2: Relations between the main AS/RS components.

Physical components:

Items, Unit-load containers and Storage / retrieval requests (model entities)

AS/RS are designed to handle storage and retrieval requests for specific items. An item represents a stock keeping unit that is stored in unit-load containers and is present in a certain quantity in the AS/RS. In order to differentiate the content of a unit-load container, each item is tagged with its stock keeping unit number. A lot more information can be included in each item. In some industrial context (e.g. Food industry), it is mandatory to retrieve the oldest product first in order to maximize shelf life. From this perspective, the item's production date of each product need also to be recorded by the system.

Retrieval requests are modeled as 6-tuples (p, ζ, ξ, x, y, z) indicating the exact location where the request will be executed. p, ζ and ξ represent the item number, aisle number and pick face (side) respectively while x, y and z represent the column, row and depth of the request. Storage requests

follow in the same form but with the replacement of *p* by the unit-load container object to be stored. More parameters can be incorporated to the requests in order to measure requests processing time, from the time they enter the warehouse to the moment they are processed.

From an implementation perspective, the simulation model is composed of programming "bricks" that represent entities and servers within an AS/RS by a set of properties and procedures. For example, the most simplistic object manipulated in an AS/RS is called an item. An item can be modeled as a programming object (or class) with a single numerical field representing its Stock Keeping Unit (SKU). More fields could be added depending on the required level of precision. Two major strengths of object-orientation are of great use in the approach. First, the fact that it is possible to imbricate objects allows, for example, to store a collection of items in a unit-load container and, furthermore, storing a unit-load container in a specific storage location of an aisle. Figure 3 illustrates the imbrication of objects that compose the AS/RS. In this Figure, we are only interested in the relationship between the objects and will discuss the procedures later. Arrows show the relationship direction and plurality (e.g. The AS/RS is composed of "1 to many" Aisles where each storage location can hold a single unit-load container, etc.). The second strength of OO modeling is the ability to directly instantiate multiple copies of an object at the initialization phase. Each instance holds its own information while reacting by the same rules as the all objects of the base class. This feature is particularly useful to model multiple aisles AS/RS.

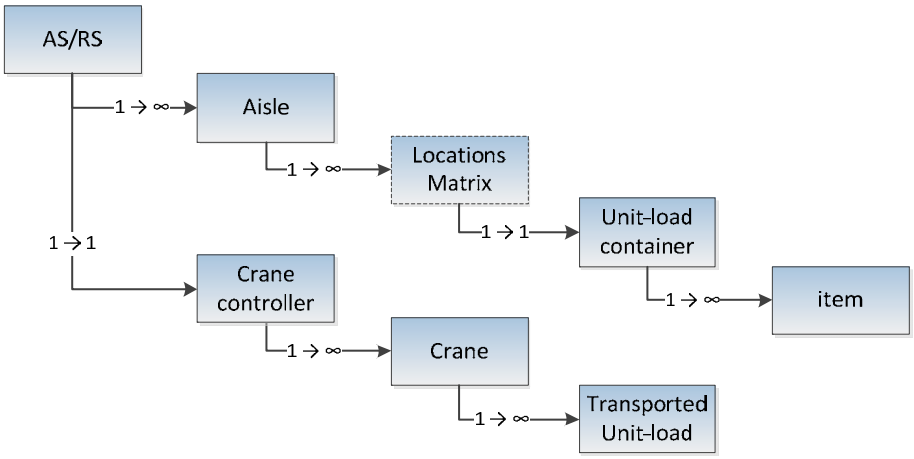

Figure 3: Hierarchy of imbrication of AS/RS objects.

Aisles and storage locations

AS/RS aisles hold unit-load containers in fixed size storage locations. From our modeling point of view, a container may represent a pallet or a bin. Each container can hold a single item (unit-load AS/RS) or a collection of items (e.g. end-of aisle AS/RS employed to handle the storage small parts). Storage locations of an aisle can be modeled as two 3-dimensions arrays of unit-load containers, one for each side of the aisle. This approach allows modeling a wide variety of storage configurations including multiple depth storage locations. Using this approach, we assume that storage locations are of the same dimensions and that any location has the capability to hold any unit-load container. Storage aisles have one or more I/O points which can be modeled by their unique coordinates in the rack.

Decisional components

Crane controller

The crane controller models the decisional part of the AS/RS. Every request that is processed by the warehouse is handled by the controller. The controller is responsible for the split of storage and retrieval requests in two independent queues and their sequencing. It is also responsible for the assignation of requests to cranes. Storage tasks represent physical unit-load containers. Changing the order (the sequence) in which storage tasks are treated usually requires a physical conveyor loop. Retrieval requests are of decisional nature and can be sequenced without physical requirements. In order to sequence storage and retrieval requests, the controller has to communicate with each aisle to evaluate its available stock. The controller also has to communicate with cranes in order to know their current location. For example, a simple yet efficient rule for task sequencing is the “shortest-leg” rule which looks for the closest location from a crane present position in order to minimize travel-time to complete a certain request. Transmission of requests may follow a push or a pull policy. The push policy implies that a block of requests is sequenced and then “pushed” to cranes queues. While this approach may require less computing, it is also susceptible to yield poorer results in terms of travel-time. As an alternative, it is also possible for the cranes to “pull” a new storage and/or retrieval request from the controller whenever a cycle is complete. In this way, it is possible to sequence requests in a dynamic and that will more likely to yield shorter travel-times. Since hybrid command is the best strategy in order to reduce travel-time while keeping good service levels

(see Eben-Chaime and Pliskin, 1997), it seems pertinent to integrate this logic in the crane model which will be discussed next.

Stacker cranes

Stacker cranes are considered as simple servers which function is to process the storage and retrieval requests received from the crane controller. Each crane has an inbound (storage) and an outbound (retrieval) queues. The storage queue represents a list of physical unit-load container objects to store in an aisle, while the retrieval queue lists the information concerning loads to retrieve. From our modeling point of view, sequencing of the requests is fulfilled by the crane controller only. Stacker cranes typically execute single or dual command cycles. A single command cycle concerns a single request while dual command cycles usually combine a single storage and a single retrieval requests in the same travelling cycle. The former approach is preferred as it minimizes travel-time by reducing the number of empty travel legs. Since storage and retrieval requests may not always be available, the hybrid command policy (to perform dual-command cycles whenever possible) can be applied and has been shown (Eben-Chaime and Pliskin, 1997) to minimize crane travel-times.

Stacker cranes move between the locations of the ASRS according to the tasks received from the crane controller. In unit-load ASRS, cranes typically move simultaneously on both axes. When both storage and retrieval requests are available, a dual command cycle consists of performing a storage operation followed by a retrieval operation. Figure 4 summarizes the crane movement logic. In this Figure, bold segments indicate that the crane is loaded with a pallet.

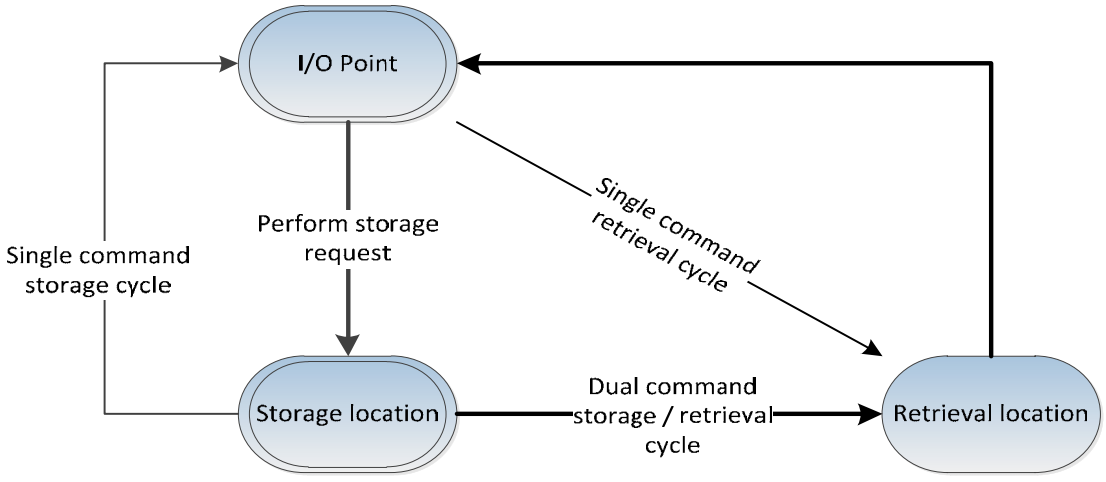

Figure 4: Model of the crane movements

Figure 4 above shows the typical movement model of an ASRS crane. Each crane is independent and follows this logic. Without loss of generality, let's assume that a crane has just completed an extraction request and thus it is located at the aisle I/O point. Depending on the availability of requests for this crane, two different paths are possible (see Figure 4). In request of both types are available, the crane will first perform a storage request, moving a load from the I/O to the storage location. Once the storage operation is completed, the crane performs an empty move to an extraction location where it retrieves a unit-load and transfers it back to the I/O point. Figure 4 also illustrates the advantage of using the dual-command approach whenever possible. In the same Figure, double framing indicate possible standby locations (i.e. crane idle).

Supply model - Inbound and outbound Interfaces

ASRS are usually linked to other systems of the distribution environment using inbound and outbound conveyors. Inbound conveyors handle goods from the production system or inbound docks. Outbound conveyors are used to carry goods from the ASRS to outbound docks where shipments can be consolidated and loaded into trucks. Conveyors are a crucial element to model if one is interested in studying the performance of other systems around the ASRS. In this case, it is often assumed that no congestion occurs on the conveyors. This assumption is plausible if inbound and outbound conveyor networks are seen as separate entities. It is also plausible to assume that goods circulating on the conveyors travel at constant speed, thus, travel-time on the conveyors is computed as a linear function of the distance and travel speed. A loop in the inbound conveyor allows sorting storage requests and, in real life scenarios, allows handling problems with the automatic reading of unit-load information (e.g. caused by a torn barcode).

Depending of the need of the study, different supply models can be applied upstream of the ASRS. In most cases found in the literature, storage requests are viewed as a list of requests generated at the beginning of the simulation. This approach is valid when the interest is focused solely on the performance of the ASRS as an independent entity. But, as expressed by Roodbergen and Vis (2009), total warehouse performance cannot be assessed only by adding up performances of all individual systems. From this point of view, it is necessary to include a supply sub-model which is responsible for the generation of storage requests in time. The proposed approach allows the modeling of one or more production centers having different setup and cycle time distributions. Using this configuration, it is also possible to model several alternatives depending on the desired decoupling point location in the production / distribution facility. Many modeling challenges appear

according to the chosen supply model. The simplest case is the pull policy where inbound and outbound flows are strongly linked: unit-loads are generated at the supply model anytime a unit-load leaves the ASRS. This approach guarantees that no space shortages will occur and can be quite useful to manage storage requests releases in time. Nonetheless, it does not seem to match appropriately real-world situations because owning an AS/RS usually means that a perfect pull policy is hardly achievable. The second scenario is the push policy, corresponding to the situation where the ASRS is the decoupling point. Goods are produced according to a predetermined production plan and, afterwards, pushed to the ASRS. This situation is more challenging because space shortages can occur, even more when production lots increase in size. From a simulation perspective, the only element that changes between the two policies is the condition set to verify before launching a production batch. In the first case, stock levels are verified after any retrieval while in the other, lots are launched following a schedule.

Discrete event simulation Engine

Discrete event simulation focuses on the asynchronous creation and execution of events in a simulated system. The coordination of the simulation is managed by a discrete-event engine. The simulation engine is similar to the one proposed by Pidd (2004), consisting of a three-phase algorithm that allows the clock to be advanced asynchronously from one event to the next. The simulation engine works with a list of events sorted by their execution time. Each time an event is executed, the system state is modified accordingly, and the simulation clock moves to the following event in the list. Sometimes, the execution of an event does not change the state of the system, but rather generates new events to be added to the list. In this case, the time at which the event will be executed is computed or simulated.

Events are classified into bounded (B) and conditional (C). B-type events are those for which the execution date can be predicted (or simulated) by the system. For example, let t be the current time at which a particular event (“travel from I/O to storage location”) is executed. Execution of this event implies the generation of a new event (“unload the pallet at the storage location”) whose execution date can be stated as $t + d$, where d is the travelling time between the I/O point and the selected location. If travelling times are modeled using probabilistic distributions, d may be a deterministic value or a variate. On the other hand, the execution date for conditional events cannot be determined in advance because they depend on a given system state. For example, it is not possible to set the execution date of an event like “Start storage operation” in advance because the event could only be executed when the following two conditions are satisfied: (1) a storage request is waiting to be

processed in the queue and (2) the crane is idle. By defining different condition sets for conditional events, it is quite straightforward to integrate specific behaviors like the hybrid command mode explained above.

In order to manage multiple objects (multiple parallel servers and entities), it is quite necessary to store these objects in arrays. In this manner, it becomes possible to maintain references to objects only by knowing their array index. This approach allows developing data structures (e.g. a single event) that are much lighter. This is an important feature, knowing that a typical simulation handles thousands of events. For example, by adding a "crane number" field in the event structure, it is always possible to know which crane is concerned by the execution of an event. The same holds for a manufacturing simulator using multiple production centers working in parallel. Table 1 shows some data structures that are required by the simulation model. These data structures are also required because the discrete event engine follows a process that doesn't have memory. In this Table, we assume that all object instances are stored in arrays and are only referenced by their ID so all stored values are of numeric nature only. As an example, each time an event occurs; the engine reads the event and advances the clock to the time of the event. Since each event concerns a specific crane or production center, the reference is passed on at each bound event concerning the same server. The same holds for the requests (storage / retrieval). These request structures transit from the controller to the cranes who execute them. Finally, production requests specify the item to produce and the required quantity. This information may depend on a pre-determined production plan or on a pull approach as stated previously.

Event structure	Requests structure	Production request structure
Time	Request arrival time	Request creation time
Event ID	Request ID	Stock keeping unit ID (SKUid)
Event Name	Request type	Quantity (Rq)
Crane Number	Stock keeping unit ID	
Production Center Number	Destination aisle number	
	Destination location	

Table 1: Sample data structures

Our three-phase algorithm works as follows. In phase A, it seeks the first event in the sorted list and the system clock is set to the execution time of the current event. In phase B, the current event is executed (i.e., the required changes to the system are applied and, if necessary, new events are added to the list). In Phase C, it verifies whether or not the system state makes it possible to execute type-C

events by testing the necessary conditions given in Tables 2 and 3. After executing each A-B-C loop, the algorithm verifies whether or not the simulation stop conditions have been met. If so, the algorithm stops; if not, it moves back to Phase A.

Tables 2 and 3 show the creation and execution logic of the simulation events in addition to state changes caused by their execution. The equations assume the following notation:

Vel_x: Crane velocity on the horizontal axis
Vel_y: Crane velocity on the vertical axis
$t_{loading}$: Time required for the crane to load a unit-load
$t_{unloading}$: Time required for the crane to unload a unit-load
T_{Now}: Current simulation time
ICL: Inbound Conveyor Length
ICV: Inbound Conveyor Velocity

A simple example is used to illustrate the algorithm's execution process. Let us assume non-empty lists of extraction and storage requests produced by the dispatcher are available at time $t = 0$, and that the event list is empty. An initialization process resets the system clock and places the crane, in an idle state, at the I/O point. Without loss of generality, the initialization process also sets the crane's last activity to "storage" so it always starts with an extraction task at the beginning of a simulation run. Since the event list is empty, the algorithm skips Phase A and Phase B and moves directly to phase C. For instance, the event "Start retrieval operation" is created if the crane is idle and if there is at least one extraction request in the queue. When this event is executed, the crane's state changes to "idle = false" and a new event, "Crane Loaded for Retrieval", is added to the events list. The execution time for this new event is the current clock time plus the expected travelling time, which can be computed based on the distance between the I/O point and the targeted extraction location chosen by the crane coordinator. Since the crane's state is "idle = false", none of the other type-C events can be executed. Thus, the algorithm moves back to phase A. The next event in the list is the "Crane loaded for Retrieval", which has just been generated. The algorithm selects this event, updates the clock to its execution time and then executes the event, which creates the "Retrieval Completed" event. The current event can then be deleted from the events list. The algorithm moves on to Phase C but cannot meet the type-C conditions. When the "Retrieval Completed" event is executed, the crane become idle again and type-C conditions will be verified in order to launch another event. In the same manner, all events are executed until a storage request is completed. In order to avoid deadlocks, discrete event engines must always have at least one event in the list.

Event Name	Event Type	Execution Conditions	State Changes	New Event Generated	New Event Execution Date	Real system behavior
Start Retrieval Operation	Conditional	1- Non-empty retrieval queue 2a- Last operation is storage OR 2b- Last operation is retrieval and storage queue is empty 3- Crane is idle	Crane is idle = false	Crane Loaded for Retrieval	$T_{Now} + Max\left(\frac{\Delta_x}{Vel_x}, \frac{\Delta_y}{Vel_y}\right) + t_{loading}$	
Crane Loaded for Retrieval	Bounded	-	Storage location is unocuppied Available stock update	Retrieval Completed	$T_{Now} + Max\left(\frac{\Delta_x}{Vel_x}, \frac{\Delta_y}{Vel_y}\right) + t_{unloading}$	Movement from current location to retrieval location Loading of pallet to retrieve
Retrieval Completed	Bounded	-	Crane is idle = true Remove request from queue	-	-	Movement from retrieval location to I/O Unloading of pallet to retrieve
Start Storage Operation	Conditional	1- Non-empty storage queue 2a- Last operation is retrieval OR 2b- Last operation is storage and retrieval queue is empty 3- Crane is idle	Crane is idle = false	Storage Completed	$T_{Now} + t_{loading} + Max\left(\frac{\Delta_{x1}}{Vel_x}, \frac{\Delta_{y1}}{Vel_y}\right) + Max\left(\frac{\Delta_{x2}}{Vel_x}, \frac{\Delta_{y2}}{Vel_y}\right) + t_{unloading}$	
Storage completed	Bounded	-	Crane is idle = true Remove request from queue Storage location occupied Available stock update	-	-	Movement from current location to I/O Loading of the pallet to store Movement from I/O to storage location Unloading of the pallet to store

Table 2: Events linked to crane activities

Event Name	Event Type	Execution Conditions	State Changes	New Event Generated	New Event Execution Date	Real system behavior
Production Center Start	Conditional	1- Production center is idle 2a- Following stock levels OR 2b- Pre-determined production plan	Production center is idle = false	For i = 1 to (Rq-1): Unit-load Completed For i = Rq: Production Completed	$T_{Now} + Setup_{SKUid} + (i * cycle_{SKUid})$	
Unit-load completed	Bounded	-	-	Unit-load arrival to AS/RS	$T_{Now} + \frac{ICL}{ICV}$	A single unit-load exits the production system
Unit-load arrival to AS/RS	Bounded	-	Enqueue new storage request at crane controller level	-	-	The produced unit-load arrives at the AS/RS
Production Completed	Bounded	-	Production center is idle = true	-	-	Production system has completed request

Table 3: Events linked to production (supply) activities

4. Possible model extensions

Using the Object-Oriented modeling approach presented in section 3, it is quite straightforward to implement extensions or variations to the model in order to study different system configurations. In this section, we give 3 examples to illustrate the possibilities of the approach.

Double-deep unit-load AS/RS

Double-deep AS/RS are based on the same concept as shown in Figure 1 with the exception that storage racks have the ability to store two unit-loads, one behind another, in each location. This double depth characteristic maximizes space utilization but, like in conventional warehouses, it requires more work in order to store or retrieve unit-loads located in the second storage layer.

It is possible to replicate double-deep AS/RS from the general model (single-depth) by performing three simple changes from the original model. First, each rack must be modeled as a 3-dimensions matrix (x, y, z), instead of the required 2-dimensions (x, y) for single-depth operation. Second, requests must now contain an indication of the depth where it must be performed. Finally, the crane movement logic must be modified in order to manage double-depth requests, as shown in Figure 5. In fact, when a crane reaches a request location and that the request target is located in the second layer, blocked by a load on the first layer, it is necessary to move the blocking load to an empty relief location. Double-deep AS/RS brings a computational challenge because the time required to store or retrieve a product becomes dependent of the state of the system. As the reader may notice, using the same approach, it is possible to manage systems having a storage depth superior to 2 levels.

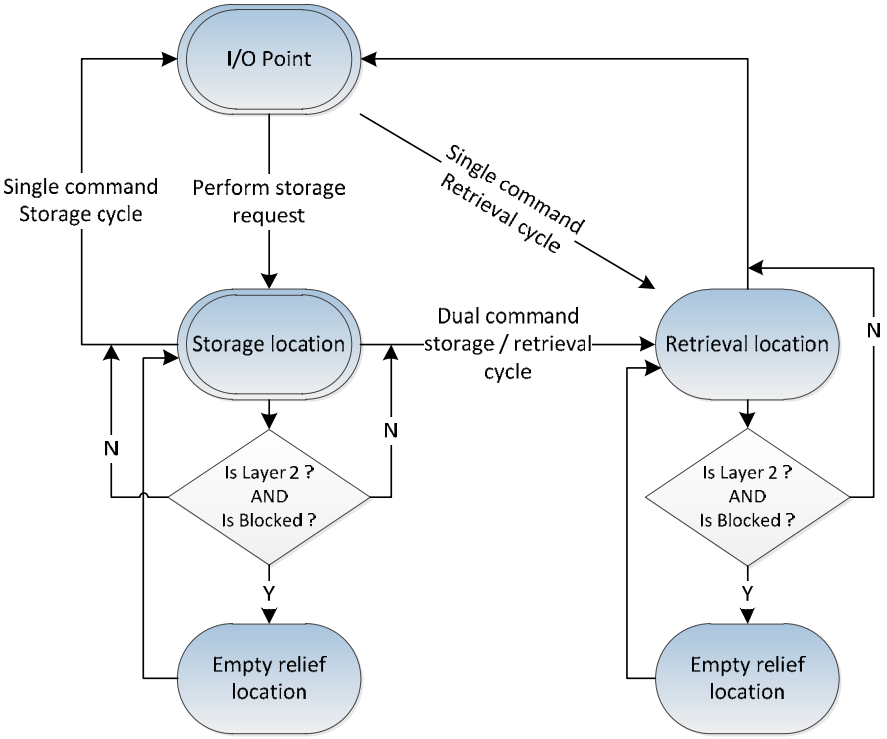

Figure 5: Changes required in the crane movement logic for the double-deep unit-load AS/RS.

Non aisle-captive cranes

In some cases, AS/RS cranes can move freely from one aisle to another by means of a rail system usually located at the end of the aisles. One of the major benefits to this approach is the fact that it allows to design a system that has less cranes than aisles. Since a great portion of the investment and maintenance costs are directly linked to cranes. It is possible to benefit substantial economies by concentrating slow moving items in an aisle which is only visited when required.

In order to mimic this kind of configuration, it is necessary to know that two cranes cannot coexist in the same aisle. To this end, it is preferable that each crane work queues (storage / retrieval) contains a maximum of one request at all times. This way, when a crane goes idle, it queries the controller which sends a request that a) minimizes the travel distance of the crane to the work location and b) ensures that each aisle contains a single crane at all times. Since each request contains information on the destination aisle and location where it should take place, no additional information is required. Finally, it is necessary to modify the travel-time formula for this particular setting, by adding an additional time component $Tr_{j,j}$, representing the time to transfer the crane from aisle j to aisle j'.

End of aisle AS/RS

The end-of-aisle configuration is often used in the less than unit-load, product to picker AS/RS context. For instance, this kind of configuration may be seen in automated libraries where bins are retrieved from the system, then processed by operators, and finally stored back into the system. In order to reproduce this kind of configuration, the general modeling approach presented in this paper may be applied very well without major changes to the framework. Conceptually, since storage racks cells dimensions are a parameter of the simulation, travel-time will be automatically adjusted to reproduce smaller locations, typically found in end-of-aisle AS/RS. Also, in order to mimic input and output operations, a few elements must be introduced into the model. These new components represent post-retrieval events, one or more processing queues and servers (pickers). In this context, when a retrieval request is completed, the selected bin goes to a processing queue in order to be manipulated by an employee (picking or putaway). After processing, the bin goes back to the controller storage queue in order to get stored back in the AS/RS. By using the object-oriented modeling approach, it is possible to easily update the quantity of each item in a bin in addition to their types.

5. Conclusions

In this paper, we have proposed a literature review of papers focusing on the study of automated storage and retrieval systems using a form of simulation. We also emphasize the fact that, concerning simulation, most studies do not explicitly detail the models used. This paper positions itself by proposing a formal modeling approach that can very well be reproduced.

The model decouples inbound and outbound flows and marks a separation between physical and decisional components of an AS/RS as well. This approach enables to model multiple scenarios, ranging from unit-load to end-of-aisle AS/RS with very few modifications.

References

1. Azadivar, F. (1984). *A Simulation-Optimization Approach to Optimum Storage and Retrieval Policies in an Automated Warehousing System*. Proceedings of the 1984 Winter Simulation Conference.

2. Bozer, Y.A. (2010). *Private communication*. September 9th.

3. Chiccholar, A.K., and Chetty, O.V.K. (1996). *Simultaneous Optimization of Control Factors in Automated Storage and Retrieval Systems and FMS Using Stochastic Coloured Petri Nets and the Taguchi Method*. International Journal of Advanced Manufacturing Technology, No. 12, pp. 137-144.

4. Eben-Chaime, M. (1996). *An integrative model for automatic warehousing systems*. International Journal of Computer Integrated Manufacturing, Vol. 9, No. 4, pp. 286-292.

5. Eben-Chaime, M., and Pliskin, N. (1997). *Operations Management of Multiple Machine Automatic Warehousing Systems*. International Journal of Production Economics, Vol. 51, pp. 83-98.

6. Fukunari, M. and Malmborg, C.J. (2008). *A heuristic travel-time model for random storage systems using closest open location load dispatching*. International Journal of Production Research, Vol. 46, No. 8, pp. 2215-2228.

7. Gagliardi, J.P., Renaud, J., and Ruiz, A. (2010). *On Storage Assignment Policies for Unit-Load Automated Storage and Retrieval Systems*. Accepted: International Journal of Production Research.

8. Graves, S.C., Hausman, W.H., and Schwarz, L.B. (1977). *Storage-retrieval interleaving in automatic warehousing systems*. Management Science Vol. 23, No. 9, pp. 935-945.

9. Han, M.H., McGinnis, L.F., Shieh, J.S., and White, J.A. (1987). *On Sequencing Retrievals In An Automated Storage/Retrieval System*. IIE Transactions, Vol. 19, No. 1, pp. 56 -66.

10. Hausman, W.H., Schwarz, L.B., and Graves, S.C. (1976). *Optimal Storage Assignment in Automatic Warehousing Systems*. Management Science, Vol. 22, No. 6, pp. 629-638.

11. Knapp, G.M., and Wang, H-P. (1992). *Modeling of Automated Storage/Retrieval Systems Using Petri Nets*. Journal of Manufacturing Systems. Vol. 11, No. 1, pp. 20-29.

12. Meller, R.D., and Mungwattana, A. (2005). *AS/RS Dwell-point Strategy Selection at High System Utilization: a Simulation Study to Investigate the Magnitude of the Benefit*. International Journal of Production Research, Vol. 43, No, 24, pp. 5217-5227.

13. Randhawa, S.U., McDowell, E.D., and Wang, W-T. (1991). *Evaluation of Scheduling Rules for Single and Dual-Dock Automated Storage/Retrieval System*. Computers and Industrial Engineering. Vol. 20, No. 4, pp. 401-410.

14. Roodbergen, K.J., and Vis, I.F.A., (2009). *A survey of literature on automated storage and retrieval systems*. European Journal of Operational Research, Vol. 194, pp. 343-362.

15. Schwarz, L.B., Graves, S.C., and Hausman, W.H. (1978). *Scheduling Policies for Automatic Warehousing Systems: Simulation Results.* AIIE Transactions, Vol. 10, No.3, pp. 260-270.

16. Van den Berg, J.P. (1996). *Class-based Storage Allocation in a Single Command Warehouse with Space Requirement Constraints*. International Journal of Industrial Engineering, Vol. 3, pp. 21-28.

17. Van den Berg, J.P., and Gademann, A.J.R.M. (2000). *Simulation Study of an Automated Storage/Retrieval System.* International Journal of Production Research, Vol. 38, No. 6, pp. 1339-1356.