WAREHOUSE SIMULATION THROUGH MODEL CONFIGURATION

Jacques Verriet, Roelof Hamberg
TNO-ESI
P.O. Box 513
5600 MB Eindhoven
jacques.verriet@esi.nl
roelof.hamberg@esi.nl

Jurjen Caarls
Dimenco
De Run 4281
5503 LM Veldhoven
jurjen@dimenco.eu

Bruno van Wijngaarden
Vanderlande Industries
Vanderlandelaan 2
5466 RB Veghel
bruno.van.wijngaarden@vanderlande.com

KEYWORDS

Warehouse systems, simulation, configurable models, sensitivity analysis, design-space exploration.

ABSTRACT

The pre-build development of warehouse systems leads from a specific customer request to a specific customer quotation. This involves a process of configuring a warehouse system using a sequence of steps that contain increasingly more details. Simulation is a helpful tool in analyzing warehouse design alternatives, but setting up a detailed simulation is too expensive early in the development process. We show that configurable simulation models can be applied early in the development process with a good cost/benefit ratio. We present a warehouse simulation model that can be configured with customer and topology information and decision algorithms. We show that the simulation results are similar to those of detailed simulations while a warehouse simulation can be configured with little effort and the simulations run fast enough to support sensitivity analysis and design-space exploration.

1 INTRODUCTION

The development of complex systems requires frequent comparisons of different architecture, design, and realization options. Many decisions have to be made, often without having all relevant information at hand. Moreover, the decisions with the highest impact have to be taken at the moment when the degree of uncertainty is also highest.

Warehouses are logistic systems that are characterized by a complex interaction between many different cooperating components. The properties of a warehouse under development are hard to predict from its components. Especially when new components or new methods of operation are involved, assumptions and reasoning are required to make accurate predictions. Moreover, the available effort to estimate any arbitrary system property is very limited.

Simulation is a common way to evaluate a warehouse's performance. Developing a simulation of a warehouse may be very time-consuming. The main reason for this is that the detailed warehouse layout is taken as a starting point for the construction of a simulation model. As an easily configurable alternative we propose the use of highly abstract building blocks to construct a warehouse simulation model, which still takes the transport configuration and main performance parameters into account.

1.1 Related Work

Simulation is commonly used for warehouse performance evaluation. Colored Petri nets have been used to simulate Automated Storage and Retrieval Systems (AS/RS) (Dotoli and Fanti. 2005; Hsieh et al. 1998). Potrč et al. present an AutoMod AS/RS simulation (Potrč et al. 2004). Gibson and Sharp (Gibson and Sharp. 1992) and Petersen (Petersen. 2000) consider simulation of a complete warehouse. They compare different batching and order-picking strategies for specific warehouse topologies. Gagliardi et al. simulate a specific high-volume warehouse (Gagliardi et al. 2007). These simulation results are typically specific for the modeled system although the modeling approach can be reused for other systems.

The simulation by Andriansyah et al. is not limited to a single warehouse (Andriansyah et al. 2011). They present a layered warehouse simulation model built from reusable components. Their approach allows them to vary the number of storage aisles and workstations in a miniload-workstation order-picking system. Although they can handle more than one warehouse, they can only handle one type of warehouse topology.

Brito presents an integrated tool to configure detailed warehouse simulations from CAD drawings and other warehouse information (Brito. 1992). This tool allows a warehouse designer to set up a detailed simulation without any programming, unless the simulation involves specific control rules.

Gu et al. provide an extensive overview of warehouse performance analysis models (Gu et al. 2010). Their overview includes many analytic and simulation models. They claim that simulation models are typically used for evaluating one design alternative, but that they are less suited for design-space exploration.

1.2 Outline

In this paper, we will address the observation made by Gu et al. (Gu et al. 2010) by introducing a warehouse

Proceedings 27th European Conference on Modelling and Simulation ©ECMS Webjørn Rekdalsbakken, Robin T. Bye, Houxiang Zhang (Editors)
ISBN: 978-0-9564944-6-7 / ISBN: 978-0-9564944-7-4 (CD)

simulation concept that can be applied in the early phases of warehouse development. This is achieved by having a highly configurable simulator that can be set up quickly, is sufficiently accurate, and has short simulation times. Our concept is similar to Brito's (Brito. 1992), but requires less information to set up and allows simpler usage of specific control rules. This means our simulation concept can be applied in the early phases of warehouse development.

The paper is organized as follows. Sections 2 and 3 describe the simulation concept with its configuration parameters and the corresponding visualization. Sections 4, 5, and 6 demonstrate the simulator on an existing retail warehouse; they explain how the simulator can be configured for the existing warehouse and compare the configured simulator to a detailed simulation performed by Vanderlande Industries. The paper ends with a reflection on the results.

2 SIMULATOR CONCEPT

A warehouse is a facility that provides temporary storage for many different products (stock keeping units, SKUs) on many physically different locations. The purpose of a warehouse is to fulfill customer orders. Fulfilling an order involves collecting a specific set of stored products and shipping them to the customer. In a typical warehouse many customer orders are handled simultaneously.

One can distinguish two main warehouse principles: man-to-goods and goods-to-man. In a *man-to-goods* warehouse, order pickers travel along a warehouse's storage locations to retrieve the products required for a customer order and putting them in an order bin, called *tote*. After the retrieval of the necessary products, the order picker travels to a consolidation workstation where the tote is dropped off and combined with other totes for the same order. Subsequently, the order picker starts working on the next tote. Note that the order picker can be human, a conveyor system, robots, or a combination of transportation means.

The other main warehouse principle is called *goods-to-man*. In a goods-to-man warehouse, the role of products and orders are interchanged: totes containing products travel towards order pickers in picking workstations where all order collecting is performed.

Our flexible warehouse simulation concept is based on the man-to-goods principle. The simulator mimics the man-to-goods principle using two types of (autonomous) entities: *totes* and *segments*. A warehouse is represented by a collection of segments that form the warehouse's transportation system. Segments are directed arcs along which the totes can travel. Each segment consists of a discrete number of sequential locations, which a tote has to visit to fulfill an order. The tote's movements are synchronized with the segment locations: in each time step, a tote can move exactly one location, i.e. from the current location on its route to the next. This means that the desired travel times are obtained by specifying the appropriate segment lengths.

Totes represent order pickers travelling to storage locations to fulfill customer orders. At several locations in the network, a tote has to stop to perform operations, such as picking a stored item or dropping off picked products at consolidation. These operations are modeled as delays for the totes.

Although our simulation concept is based on the man-to-goods principle, it can also be applied to capture a goods-to-man warehousing process. This is achieved by interchanging the role of storage and picking locations.

Our simulation concept consists of a number of configurable steps divided over two main phases: *configuration* and *simulation*. The phases and the steps are visualized in the state machine in Figure 1: the configuration steps are shown on the left; the simulation steps on the right.

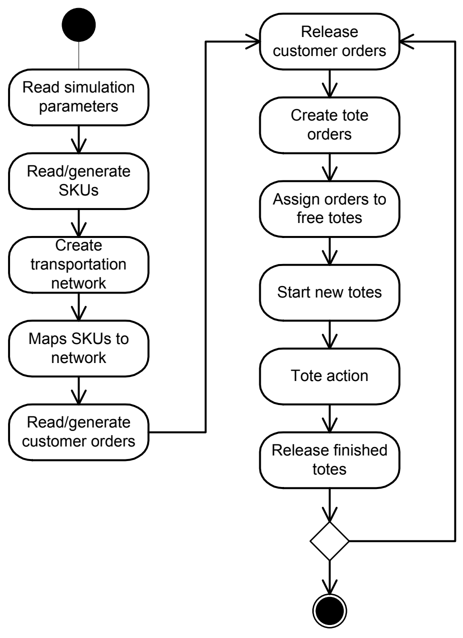

Figure 1: Simulation Concept

2.1 Simulation Configuration

The configuration phase provides the flexibility of our simulation concept. It consists of five steps: (1) reading a parameter file; (2) reading or generating stock keeping units (SKUs); (3) creating a transportation network; (4) allocating SKUs to the network; and (5) reading or generating customer orders. Each of these steps is explained in more detail.

The configuration phase starts by reading a parameter file. This parameter file contains settings for all subsequent steps. For instance, it specifies the files to be read in the following configuration steps. However, it also specifies the number totes in the system and the maximum volume and weight of the totes.

The second configuration step involves either reading a SKU file or generating a collection of SKUs. The SKUs describe the products stored in the warehouse. This is specified in the parameter file: it specifies either the SKU file to be read or distributions for the dimensions, weights, and expected sales of the SKUs.

The third step involves the creation of a transportation network. There are two ways in which a transportation network can be created. One is a layout file: the parameter file read in the first step includes a layout file to be read. A layout file consists of a list of segments. Each segment is described by a number of parameters. These include the 3D coordinates of its start and end point; its *length*, i.e. the number of locations; its *storage capacity*, i.e. the number of SKUs that can be stored per location; its *width*, i.e. the number of totes that can be at one location at the same time; its *operation type*, i.e. the type of operation performed at the segment (e.g. input, output, picking); and a distribution of the *operation times*, i.e. the time needed for these operations. From the collection of segments, a transportation network is created. If the end location of a segment is adjacent to the start location of another, then they are connected in the transportation network.

The second way to create a transportation network involves a number of pre-defined templates, which facilitates the generation of the transport network. The layout does not have to be created by hand, but rather is generated on the basis of some template-specific parameters, like dimensions and capacities. These parameters are included in the parameter file.

The fourth configuration step takes the collection of SKUs and allocates them to the network segments with a positive storage capacity. This allocation may involve SKUs to be allocated to more than one storage location. By spreading fast-moving products, the picking load can be divided over the warehouse.

The fifth and last configuration step involves reading a customer order file or generating a collection of customer orders depending on what was specified in the parameter file. If an order file is generated, then the expected sales of the SKUs are taken into account as well as an order size distribution.

2.2 Simulation Steps

Our simulator is a time-triggered simulation which repeats the six simulation steps in Figure 1. Each time starts by (1) releasing customer orders and (2) creating tote orders for each of the released customer orders. The creation of tote orders considers the maximum volume and weight of a tote and the SKU storage locations.

The simulation continues by (3) assigning open tote orders to idle totes and computing the corresponding routes for the totes to follow, and (4) introducing the totes into the system.

The most important step of the simulation involves (5) the operation of the totes. If a tote has to perform an operation, then it waits at its current location until the operation's time has passed. Otherwise, it will negotiate with the next location on its route. This involves claiming the network location by a tote. After all totes have made their claims, each network location grants some of the totes' claims. Finally, the totes with granted claims move to the next network location.

The last simulation step involves (6) releasing the totes that have finished their tote order route. These six steps are repeated until all orders have been fulfilled.

2.3 Behavioral Parameters

In the first step of the simulation configuration, a parameter file is read. This parameter file includes two types of parameters: structural and behavioral parameters. The structural parameters include the number of totes and their properties, the SKU and order files, and the distributions for the generation of SKUs and customer orders.

Many of the simulation steps (see Figure 1) involve the assignment of scarce resources. For instance, SKUs are allocated to storage locations, customer orders are assigned to idle totes, and totes compete for network locations on their routes. Decision algorithms handle the assignment of these scarce resources. For each type of decision algorithm, a generic Java interface has been defined. For each of these interfaces, there is a library of Java classes that implement the interface and provide a decision algorithm.

The classes in these libraries can be selected and instantiated at run time. These behavioral parameters are read from the simulator's parameter file. Using Java's introspection, the corresponding classes get instantiated for usage during simulation. A similar approach has been applied by Verriet and van Wijngaarden (Verriet and van Wijngaarden. 2012) for their reference architecture for warehouse control.

The Java interfaces allow easy inclusion of system-specific functionality. If a behavior library does not contain the necessary functionality, then one can create a new Java class that implements the corresponding interface, add it to the library, and specify its usage in the simulation's parameter file.

3 VISUALIZATION

Besides the simulator described in the previous section, we have also developed a 3D visualization that allows the warehouse designer to validate a warehouse design visually, the warehouse behavior in particular. The visualization is a C++ application using OpenScene-Graph. The simulator and the visualization application communicate by sending messages to the visualization over a UDP interface.

Figure 2 shows an example of the simulation's visualization. It shows a Zone Picking System (ZPS). ZPS is a man-to-goods warehouse concept where storage is organized in zones. To fulfill orders, totes visit a number of zones where operators take items from the zones' local storage and put them in the totes.

ZPS is one of the pre-defined templates of our simulator; the system in Figure 2 was created by specifying the number of aisles, the number of zones per aisle, the zones' buffer lengths, and several other ZPS-specific parameters.

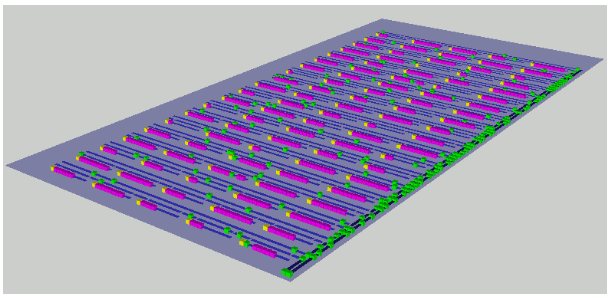
Figure 2: Zone Picking System (ZPS) simulation

4 COMPACT PICKING SYSTEM

In this section, we will demonstrate the application of our simulation concept to an existing retail warehouse. Figure 3 shows a schematic of this warehouse. It is a so-called Compact Picking System (CPS). The system consists of three main components: five *miniload* storage aisles, a main *conveyor loop*, and three picking *workstations*.

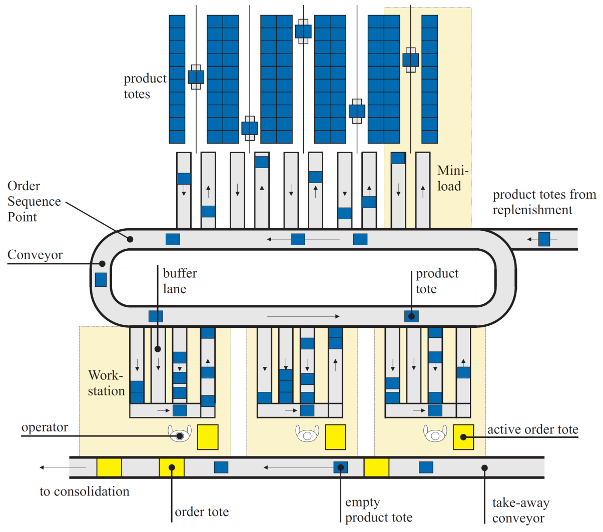

Figure 3: Compact Picking System (CPS)

4.1 Miniloads

The top of Figure 3 shows the five miniloads. Each miniload consists of a storage rack with totes and a crane that can store and retrieve totes. The cranes in Figure 3 can handle up to four totes simultaneously. The outer miniloads have a dual function in the warehouse: they have to supply not only to the CPS workstations, but also to six ZPS workstations (not shown in Figure 3). This effectively means that the retrieval capacity of the outside miniloads is smaller than that of the other three miniloads.

At the start of each retrieval cycle, a miniload determines which totes it will retrieve. This selection is based on the age of the corresponding customer orders and the number of totes underway to the different workstations. The miniload will select the oldest four totes for retrieval, unless the corresponding orders are too young. Each workstation has *pipeline* capacity, i.e. the maximum number of totes underway to the workstation. This pipeline is used to avoid that a workstation runs idle or its input buffers overflow. If a workstation's pipeline is full, then a miniload will not retrieve a tote for that workstation. Moreover, part of a workstation's pipeline is reserved for the oldest order for the workstation. Totes for younger customer orders will not be selected for retrieval if the non-reserved pipeline is full. This means that a miniload may retrieve fewer than four totes per retrieval cycle.

4.2 Central Conveyor

After totes have been retrieved from a miniload, they travel via the central conveyor to a picking workstation. The central conveyor is shown in the middle of Figure 3. This conveyor loop has a sequencing point that is used for releasing customer orders. In the CPS warehouse, totes for at most nine customer orders are retrieved simultaneously. If the last tote of a customer order has passed the sequencing point, then totes of younger customer orders cannot overtake it any more. At that moment, a new customer order is released for retrieval by the miniloads.

4.3 Picking Workstations

Via the central conveyor, totes travel to a picking workstation. There are three such workstations and they are shown at the bottom of Figure 3. At a picking workstation, a human operator takes items from the tote. After this operation, the tote returns to a miniload.

Each workstation is working on one customer order simultaneously; this is the oldest customer order assigned to the workstation. Its three input buffers allow totes of different customer orders to be buffered simultaneously while ensuring that the totes of the oldest customer order are handled consecutively. At any time, totes for three different customer orders can be underway to a workstation. After all totes of one of these three customer orders have passed the central conveyor's sequencing point, then a new order for the workstation can be started.

The CPS warehouse has three picking workstations. Two workstations are operated permanently; the third is an overflow workstation that is operated only if sufficiently many totes have been collected in its input buffer lanes. If X totes have been collected in the overflow workstation's input buffers, then an operator handles exactly X totes and then stops picking until again X totes have been collected.

The distinction between the workstations is also used when retrieving totes from the miniload. Totes for regular workstations have priority over totes for the overflow workstation. This means that a miniload will retrieve a tote for the overflow workstation only if it cannot retrieve a tote for the regular workstations.

5 CPS SIMULATION MODEL

As explained earlier, our simulation concept mimics a man-to-goods principle. The described CPS warehouse does not comply with the man-to-goods principle, but with the goods-to-man principle. To have our simulation model the goods-to-man principle, the role of storage and picking has to be reversed. In other words, in the CPS simulation model, the storage is modeled in the picking workstations and not in the miniloads.

5.1 Transportation Network

Figure 4 shows the transportation network of the simulation model. The picking workstations are shown on the left, the central conveyor loop in the middle, and the miniloads on the right. Since the totes travel over the main conveyor loop in a counter-clockwise direction, the overflow workstation is the one furthest from the miniloads.

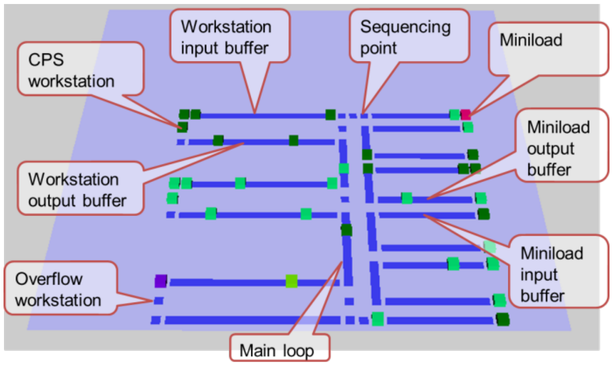

Figure 4: CPS Simulation

If one inspects Figures 3 and 4 carefully, then one can observe a few differences. The miniloads are modeled as two segments, a retrieval segment connected to a miniload output buffer and a storage segment connected to a miniload input buffer. Each tote enters the system via a miniload retrieval segment, visits one workstation (to pick one product), and leaves the system via a miniload storage segment. The miniload storage and retrieval times are modeled as operation times of the input and output segments.

Another difference in the layout involves the picking workstations. The warehouse (in Figure 3) has three input buffers per workstation. The simulation model (in Figure 4) has only one input buffer per workstation. This single input buffer has a width of three, meaning that it can hold three totes in each of its locations.

5.2 CPS-Specific Behaviors

To obtain the warehouse behavior described in the previous section, two CPS-specific behaviors have been implemented. At the end of the workstations' input buffers, there is a segment that holds the totes awaiting picking. In the simulation, these totes claim the picking location and the picking location may select one of them. To ensure that totes of one customer are handled consecutively, a specific behavior has been implemented and added to the simulator's corresponding behavior library. This behavior randomly selects a tote from the active customer order. If the active customer order has not been completed, and no tote for this order is waiting, then it will not allow any tote to go to the workstation's picking location. In other words, the picking location waits for the remaining totes of the active customer order before handling totes of younger orders.

This new behavior also takes care of the behavior of the overflow workstation: the picking location of the overflow workstation will select a tote only if X totes await picking. Then it will select totes in the same manner as the regular workstation, with the difference that it will (sequentially) select exactly X totes and then wait until X totes are waiting again.

Another specific behavior has been implemented for the selection of orders to be started. As explained in the previous section, totes for the regular workstations have priority over those for the overflow workstation. Moreover, a tote for the oldest order will be retrieved only if the corresponding workstation's pipeline is not full and totes for younger customers will be retrieved only if the non-reserved pipeline is not full. These rules have been combined in a Java class implementing the generic tote selection interface.

For the other decision algorithms, a pre-defined behavior from the simulator libraries sufficed. A movie of a running CPS simulation is available on-line (Verriet. 2012).

6 COMPARISON

The simulation model described in the previous section has been validated using real-life SKU and customer order information of the modeled retail warehouse. This information has been the input of a detailed AutoMod simulation model and our simulation model. The AutoMod model was constructed by Vanderlande Industries to validate a warehouse design; in other words, it has not been used during the early phases of warehouse development.

To compare the simulators, we have used twelve scenarios used for design validation using the AutoMod simulation. Several parameter values are varied; these include the size of the workstations' (reserved and non-reserved) pipelines, the workstations' input buffer lengths and the operators' picking times.

We have compared the scenarios using a set of orders for one peak day: 6,856 order lines divided over 156 orders. In the simulation, each order line is handled by one tote and circa 100 totes are active in the CPS system simultaneously (depending on the parameter values).

Figure 5 compares the simulated times for the different scenarios. Figure 6 provides a comparison of other performance criteria. From top to bottom, it compares the number of totes per workstation, the workstations' utilization, and the occupancy rates of the input and output buffers of the workstations and miniloads.

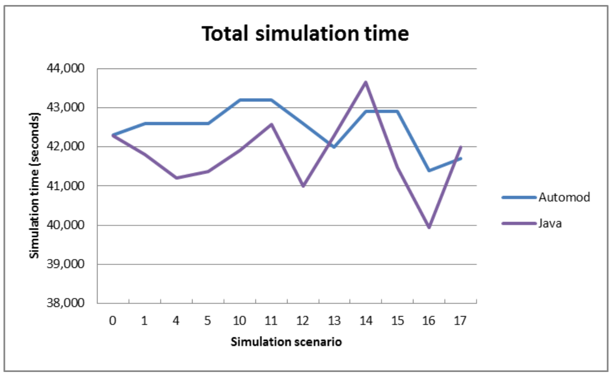

Figure 5: Comparison of Simulation Times

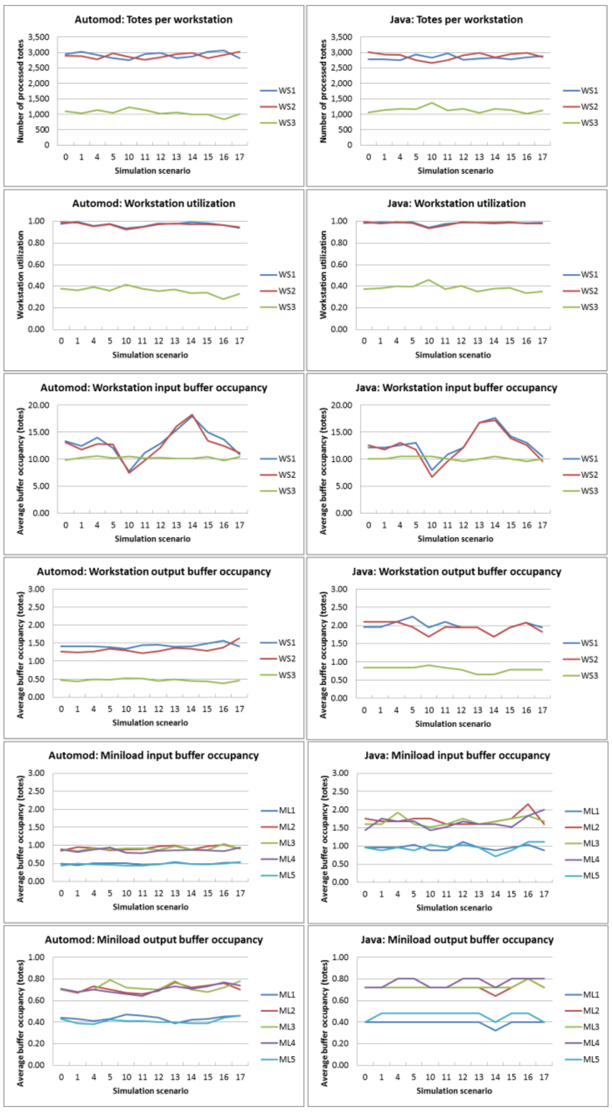

Figure 6: Simulation Comparison

The graphs in Figures 5 and 6 show that most of the performance measures of our simulation are similar to that of the detailed AutoMod simulation; most differ only a few percent. Larger differences can be seen for the occupancy of the input buffers of the miniloads and the output buffers of the workstations, i.e. the segments leading towards the central conveyor loop. These larger differences are (probably) due to the difference in claiming a window on the central conveyor loop; our Java simulator uses a simpler algorithm than the AutoMod simulation. This is due to the fact that our simulator considers the central conveyor as a collection of independent segments instead of a separate entity.

The outcome of the simulations shows a similar performance. However, there is an important difference. Our Java simulation model can be set up much quicker than the detailed AutoMod simulation. Because the transportation network of this CPS warehouse consists of only 56 segments, a first transportation network is set up within an hour. More complex transportation networks may take a few days to construct, but the corresponding visualization supports this by providing visual feedback.

The main effort in setting up the simulation involved the implementation of the warehouse-specific behaviors. Two Java classes had to be implemented in order to obtain the desired strategies for miniload retrieval and tote movement (in particular at the picking workstations). This involved dozens of lines of code, and took several hours to implement and test. Again the visual feedback provided by the visualization is very useful in obtaining the desired system behavior.

Besides the fact that setting up the simulation can be done within a few days, the simulations themselves require little time. Without visualization, the simulation of a peak day takes circa 38 seconds on a Dell Latitude E4300 laptop with an Intel P9600 processor and 4 GB of RAM running 32-bit Windows 7 SP1. This means that the simulation runs more than 1,000 times real-time, because completing the customer orders of the peak day takes about 42,000 seconds (see Figure 5).

The quick simulation setup and the small simulation times allow a warehouse designer to test a warehouse's sensitivity to specific parameter changes. The effect of changing a single (e.g. numerical) parameter can be tested within a few minutes. The fast simulations even support the exploration of a part of the design space. This has been done by Reehuis and Bäck, who have applied an evolutionary algorithm to optimize the performance of a ZPS warehouse modeled with an earlier version of our warehouse simulator (Reehuis and Bäck. 2010). Their (automated) optimization involved more than 20,000 simulation runs using different parameter settings including variations of the system layout and warehouse behaviors.

7 CONCLUSION

In this paper, we have described a warehouse simulator that can be used during the early phases of warehouse development when the most important design decisions are to be made. This is due to the configurability of our simulation concept. Its configuration parameters include SKU and order files, a warehouse network topology, and a variety of structural and behavioral parameters.

A case study of an existing retail warehouse showed that the configurable warehouse simulator provides similar results as a detailed simulation, which is typically only used in the later stages of warehouse development.

In contrast to detailed simulations, our warehouse simulation can be set up quickly: configuring the simulator takes a few days. The corresponding visualization provides visual feedback to the warehouse designer regarding the correctness of parameter choices. Setting up simulations requires little time and the same goes for the simulations themselves. Since a simulation run typically takes less than one minute, a warehouse designer is able to perform a sensitivity analysis and even perform (automated) design-space exploration.

ACKNOWLEDGEMENTS

This work has been carried out as part of the FALCON project under the responsibility of the Embedded Systems Institute with Vanderlande Industries as the carrying industrial partner. This project is partially supported by the Netherlands Ministry of Economic Affairs under the Embedded Systems Institute (BSIK03021) program.

REFERENCES

Andriansyah, R.; W.W.H. de Koning; R.M.E. Jordan; L.F.P. Etman; J.E. Rooda. 2011. “A process algebra based simulation model of a miniload-workstation order picking system.” *Computers in Industry* 62, 292-300.

Brito, A.E.S.C. 1992. “Configuring Simulation Models Using CAD Techniques: A New Approach to Warehouse Design.” PhD Thesis, Cranfield Institute of Technology, Cranfield, UK.

Dotoli, M.; M.P. Fanti. 2005. “A coloured Petri net model for automated storage and retrieval systems serviced by rail-guided vehicles: A control perspective.” *International Journal of Computer Integrated Manufacturing* 18, 122-136.

Gagliardi, J.P.; J. Renaud; A. Ruiz. 2007. “A simulation model to improve warehouse operations.” *Proceedings of the 2007 Winter Simulation Conference* (Washington, DC, Dec. 9-12). IEEE, Piscataway, NJ, 2012-2018.

Gibson, D.R.; G.R. Sharp. 1992. “Order batching strategies.” *European Journal of Operational Research* 58, 57-67.

Gu, J.; M. Goetschalckx; L.F. McGinnis. 2010. “Research on warehouse design and performance evaluation: A comprehensive review.” *European Journal of Operational Research* 203, 539-549.

Hsieh, S.; J.S. Hwang; H.C. Chou, 1998. “A petri-net-based structure for AS/RS operation modelling.” *International Journal of Production Research* 36, 3323-3346.

Petersen II, C.G. 2000. “An evaluation of order picking policies for mail order companies.” *Production and Operations Management* 9, 319-335.

Potrč, I.; T. Lerher; J. Kramberger; M. Šraml. 2004. “Simulation model of multi-shuttle automated storage and retrieval systems.” *Journal of Materials Processing Technology* 157-158, 236-244.

Reehuis, E.; T. Bäck. 2010. “Mixed-Integer Evolution Strategy Using Multiobjective Selection Applied to Warehouse Design Optimization.” In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation* (Portland, OR, Jul. 7-11). ACM, New York, 1187-1193.

Verriet, J. 2012. Falcon Compact Picking System (CPS) simulation. http://www.youtube.com/watch?v=DcJs3ALbDDg.

Verriet, J.; B. van Wijngaarden. 2012. “A Reference Architecture Capturing Structure and Behaviour of Warehouse Control.” In *Automation in Warehouse Development*, R. Hamberg and J. Verriet (Eds.). Springer, London, 17-32.

AUTHOR BIOGRAPHIES

JACQUES VERRIET received his MSc and PhD degrees in computer science from Radboud University Nijmegen and Utrecht University, respectively. In 1998 and 1999 he worked as a consultant at the Centre for Quantitative Methods in Eindhoven. He joined Siemens VDO Automotive in 2000 to work as a software engineer/researcher on car navigation. Since 2006 he works as a research fellow at the Embedded Systems Institute, which has become a department of TNO in 2013. His research areas of interest include model-driven system development and system-level control.

ROELOF HAMBERG received his MSc and PhD degrees in Physics from the Universities of Utrecht and Leiden, respectively. He joined Philips Research Laboratories in 1992 to work in the field of perceptual image quality modeling and evaluation methods. In 1998 he joined Océ-Technologies B.V. as a developer of in-product control software and digital system architect later on. Subsequently he became departmental manager, first research, later in product development. Roelof Hamberg joined ESI in 2006 as a research fellow. His research interests are easy specification, exploration, simulation, and yet formal reasoning about system behavior, and systems architecting in general.

JURJEN CAARLS received his MSc degree in Applied Physics from Delft University of Technology. His MSc thesis on 'Fast and Accurate Robot Vision' for the RoboCup robots of the Dutch Soccer Robot team Clockwork Orange won the award of the best MSc thesis of the Applied Sciences faculty in the year 2001. He received his PhD degree in 2009 from Delft University of Technology on camera pose estimation and sensor fusion for Augmented Reality; he worked on robust distributed warehouse control in the FALCON project and is currently Software and Algorithms architect at Dimenco B.V.

BRUNO VAN WIJNGAARDEN received his Master’s degree in Electrical Engineering from the Eindhoven University of Technology in 1986. He has 20+ years of experience in logistics and material handling and joined Vanderlande Industries in 2000. In his current position as system architect he is responsible for system design, market analysis and requirements definition for product development.