2018

Pull-driven Order Fulfillment in Distribution Centers

Dima Nazzal
Fortna Inc., dima@dimanazzal.com

Shankar Shanmugasundaram
Fortna Inc., shankarrs@fortna.com

Sung Kim
Fortna Inc., sungkim@fortna.com

Russell D. Meller
Fortna Inc., russmeller@fortna.com

Pull-driven Order Fulfillment in Distribution Centers

Dima Nazzal
Fortna R&D
Fortna Inc.
Reading, USA
dima@dimanazzal.com

Shankar Shanmugasundaram
Product R&D
Fortna Inc.
Reading, USA
shankarrs@fortna.com

Sung Kim
Fortna R&D
Fortna Inc.
Reading, USA
sungkim@fortna.com

Russell D. Meller
Fortna R&D
Fortna Inc.
Reading, USA
russmeller@fortna.com

***Abstract*— One of the challenges facing e-commerce fulfillment is the rigidity of the wave management system that often dictates a fixed batch size that must be processed before the next wave of orders is released. As a result, operation managers usually push multiple waves, which lead to accumulation at bottleneck resources, and poor synchronization at sortation engines. We demonstrate the power of operating an order fulfillment engine using the "pull" paradigm. We hypothesize that a pull-driven order fulfillment can meet the same throughput requirements achieved by the push-driven wave-based order fulfillment, but with higher speed, higher service levels, more level resource utilization, and smaller hardware investment cost. To that end, we develop analytical throughput models of a zone picking operation, followed by a sortation operation to separate and consolidate the orders before packing. We develop an algorithm for determining the picking batch size based on the available capacity at each process, and an algorithm for determining the maximum order threshold to maintain in the system before authorizing release of orders towards picking. We use a discrete-event simulation model to validate our approach and measure the performance differences between pull vs. push control.**

Keywords—pull framework, batch picking, zone picking, put walls

I. INTRODUCTION

The demands of e-commerce and omni-channel distribution center fulfillment is increasing the need for innovative hardware and software solutions that can adapt swiftly to macro demand pattern changes as well as seasonal and daily volume peaks. Some of these challenges presented by e-commerce in omni-channel facilities can be thought of as mass customization; large variety of SKUs, high and fluctuating volume of orders that consist of few lines and few units. In addition, customer expectations demand high service levels such as shortened delivery times and free shipping. That, coupled with the increasing difficulty to maintain high resource utilization, and high workers productivity present dueling objectives. To face these challenges, distribution center managers seek hardware and software options that can maximize productivity, speed, and service levels.

Warehouse Execution Systems (WES) is the newest iteration in the supply chain/warehouse execution software world, promising to take on these order fulfillment challenges. The promise of WES is to have a system with real-time continuous monitoring of labor status, order status, and resource status. With this level of visibility, algorithms that can dynamically optimize decisions that govern the movement and handling of orders can be built into the WES, and hence provide the flexibility to maintain continuous flow in the DC despite the peaks and valleys in order volumes and the changing order profiles.

One of the challenges facing e-commerce fulfillment is the rigidity of the wave management system that often dictates a fixed batch size that must be processed before the next wave of orders is released. As a result, DC operation managers usually push multiple waves, which lead to accumulation at bottleneck resources, and poor synchronization at sortation engines. In this paper, we demonstrate the power of operating an order fulfillment engine using the "pull" paradigm, which is enabled with the WES capability to provide continuous visibility on the status of resources and units of flow.

We show that a pull-driven order fulfillment can meet the same throughput requirements achieved by the push-driven wave-based order fulfillment, but with higher speed, higher service levels, more level resource utilization, and smaller hardware investment cost. To that end, we developed analytical throughput models of a zone picking operation, followed by a sortation operation to separate and consolidate the orders before packing. We develop an algorithm for determining the picking batch size based on the available capacity at each process, and an algorithm for determining the maximum order threshold to maintain in the system before authorizing release of orders towards picking. We used a discrete-event simulation model to validate our approach and measure the performance differences between pull vs. push control.

Our experiments validated our hypothesis and showed significant increase in fulfillment speed, and smoother flow (leading to steadier utilization of resources). We will present our conceptual model of a pull-driven order fulfillment, and the performance comparison results.

A. System Description

The system we study has two main processes, conceptually illustrated in Figure 1:

1. **Zone picking system**: Multiple units for multiple orders are batch-picked into the same container. Considerable productivity gains are achieved by batch picking because a pick location is visited once during a wave. On the other hand, order integrity is not preserved during picking and hence the need for a second process for sorting container contents to consolidate order units.

2. **Put Wall sortation system**: A put wall is a series of openings or compartments known as cubbies. One side of the put wall is staffed by an operator who puts units that belong to a

specific order into an assigned cubby. The other side of the put wall is staffed by an operator who packs a complete order. The number and size of cubbies vary depending on order cubic size and the number of orders that needs to be processed simultaneously. Most often, there are multiple put walls in a DC. Figure 2 shows a picture of a put wall used for a Direct-to-Consumer order consolidation process.

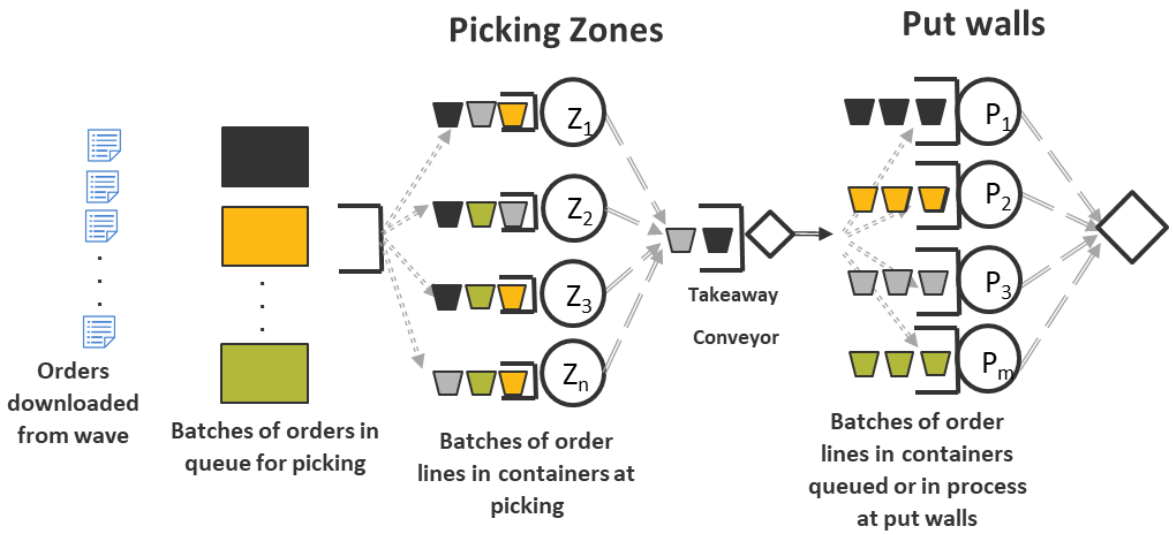

Fig. 1. Zone Picking to Put Walls Processes. Orders in a wave are batched to improve picking efficiency. Batches are released to pick zones for picking. Containers carrying the items belonging to the same batch are sent to a specific put wall. Items belonging to the same order are emptied into a specific put wall cubby. When the order items have been consolidated, the order is packed out and the cubby is now available for another order.

Fig. 2. A put wall. Each bin is dedicated to an order. Totes arrive carrying mixed SKUs. When an operator scans an item, the light under the bin where the item needs to be put lights up.

B. Basics of Batch Picking to Put Wall Process

When orders are released to picking, units are picked from multiple zones in the forward pick area into multiple containers. Containers from multiple zones are delivered to the put wall. The put wall operator starts removing units from the containers, scanning them, and putting them into the destination cubby until the container is empty. It is important to distinguish that this is a batch-by-SKU operation as opposed to the more familiar batch-by-order operation; SKUs shared by multiple orders within a wave are picked together into the same container then separated at the put wall. A Warehouse Execution System (WES) usually directs the operator to the correct cubby using a wide variety of technologies such as put-to-light or put-to-voice. Once all the required units for an order have been put into the designated cubby, the order is ready to be packed by the operator on the other side of the put wall.

Put Wall Advantages

Order and SKU profiles and the picking methodology dictate whether there is a business case for *installing* put walls in a DC. Generally, put walls are beneficial for environments dominated with e-commerce multi-line orders and order volumes that are higher than feasible for a discrete or cluster picking operation but lower than financially justifiable for a unit sorter conveyor system. Put walls require lower capital investment compared to conveyor-based unit sorters. When coupled with an intelligent WES, putting productivity and order sortation accuracy can be enhanced substantially. An argument can also be made for put wall movability, scalability, and flexibility.

Put Wall Drawbacks

The main drawbacks that we have observed in systems with a business case for put walls are operational challenges that manifest themselves in long and highly variable dwell times of orders in cubbies, erratic resource utilization patterns, and long queues of containers at the put walls. The underlying causes for these effects are the result of difficulty in synchronizing the arrival times of units for the same order. When the lines of an order are picked from different zones, they arrive to the put wall in separate containers, at different times, which results in longer dwell time of the order in its cubby. The long dwell times keep the cubbies from being turned and reused for other orders, which might create a queue of containers in front of the put wall, or a delay in the release of the next wave's orders, which leads to idle resources when there is work to be done.

Oversizing the Put Walls: System designers mitigate this effect by oversizing the put walls to create a buffer. That might solve the container accumulation problem but does not address the long order cycle times and service level implications. Additionally, larger put walls result in more travel from container to cubby and so forth, which reduces the putting productivity, especially when the WES or WCS lack the intelligence to assign cubbies to orders in a way that minimizes operator travel.

Installing a Wave Bank: Another design solution to mitigate the synchronization problem is adding a wave bank, which is essentially a central buffer to which containers are pushed after being picked. Release of containers from the wave bank improves the speed of order consolidation. The downside is the additional investment in the hardware and space required for the wave bank. Figure 1 shows a picture of a wave bank

Fig. 3. Wave Bank. Totes are waiting at the wave bank ready to be released to the put walls for sortation

Batch Size Effect

A batch is the set of orders sent to the same put wall within the same wave. A wave has multiple batches of orders. Synchronization between picking and sorting becomes harder to achieve when the number of orders in a wave or in a put wall batch increases. As the number of released orders increases, picking cycle time - which depends on the number of orders in a batch and the number of simultaneous batches- increases. Longer picking time increases resource utilization but results in longer queues at picking, and hence increases the variability in the arrival process of containers to the put wall process, which results in accumulation, that may lead to double-handling of containers due to space constraints, and longer cycle times to empty out the containers.

II. Proposed Solution: Pull-driven Order Release Control

Releasing waves on a set schedule or based on a wave size threshold is the prevalent control mechanism currently utilized in most DCs. Orders are assigned to a wave (sometimes a wave has specific characteristic such as a shipping cutoff time.) When the number of orders in a wave reaches a specific threshold or when the time between wave releases has been reached, the wave is released to picking. At that time, the orders are assigned to batches; a batch of orders is sorted at the same put wall, and hence lines belonging to the same batch can be picked together into the same container, if these lines are in the same pick zone.

Our solution proposes adapting methods from production lines operated under the pull control philosophy, in which release of jobs is *authorized* based on downstream processes status. Despite the complexity of setting the "optimal" number of orders to maintain in the system, we posited that the controlled release of order batches to picking, and hence to the put walls would level the flow, reduce queueing of containers at the put walls, and starvation of resources. Optimizing the pull threshold is not trivial even in simple flow lines. Kanban, which aims to optimize the work-in-process level at each station, and CONWIP, which sets an overall WIP level for the system, are the most popular. The CONWIP policy discussed in [1] is popular for the insights it provided, its simplicity when implemented using a Mean Value Analysis approach, and its robustness. In its basic form, the CONWIP framework applicability is limited to serial production lines with a single station at each process and one product type. In our zone-picking to put walls system, we violate all these assumptions. We have multiple zones sending jobs to multiple put walls. Each pick zone has multiple operators. The orders have different properties and processing times depending on the order size. Moreover, the unit of measure in the flow changes from a container in which multiple lines are processed together at picking and during travel, but the unit of measure that exits the system is an order. Additional challenges for designing a pull approach for controlling the flow in this system are:

1. The smallest unit of measure for release is most often not one order but a batch of orders. Otherwise, the capacity of the picking process would not be able to handle the demand. The just-in-time literature often advocates for using a batch of one job, releasing one job at a time, and reducing set up time to handle smaller serial batches. In a manual picking system, the set up time for a batch is the walking time for a picker to the location of the SKU. Automation can help with reducing this setup time but it is expensive. Additionally, releasing one order at a time would largely increase the number of containers flowing in the system, which would overwhelm the conveyor system, and likely cause jams, blocking, and recirculation.

2. The dynamic nature of the environment; varying demand especially in e-commerce Direct-to-Consumer operations; and varying capacity that depends on labor

availability, presents a challenge for setting control parameters that need to be revised based on demand and capacity conditions on a given day.

There is research that analyzes the performance of a CONWIP system with multiple products through a network of resources, or applying CONWIP to batch processes. We did not find any research publications that model a pull process with a dynamic capacity impacted by the "job" characteristics, where the job is split into multiple servers (pick zones with multiple operators in each zone), then re-assembled at another multi-server station (put walls). This type of application requires adjustments of the operational pull parameters to achieve the benefits of pulling the work. Neither did we find any work that applies the concept of pull, whether through Kanban or CONWIP, to a warehousing environment or the order-fulfillment process. The most recent relevant review is published in [2].

The objective of our work is twofold: (1) Develop a model for setting the pull system parameters that meets the throughput requirements using the available resources with minimum queueing, and (2) Use discrete-event simulation to quantify the performance gains of operating this type of system using the settings we developed in (1) and compare to a traditional push approach. Therefore, a practical and implementable solution needs to be dynamic, robust to handle deviations from assumptions, and computationally efficient.

A. A Pull Framework

The pull framework is illustrated in Figure 4, where we have a virtual queue of orders in a wave. When a wave is released, the orders are grouped into virtual batches in each zone. Some of these batches are available to pickers (i.e., have been released), and some are released when the pull signal is triggered. An available picker is assigned the next batch (or batches) to pick. A batch now is assigned to a container, and the picker executes the pick. Upon completion the container is sent to the put walls. Containers are emptied into the order cubby. When all the lines of an order are assembled at the put wall, the packer empties the cubby, and the order "exits" the system.

1) Pull parameters:

The pull-driven flow heuristic determines the values of three parameters every time a wave is dropped into the system. These parameters are:

- X: is the number of orders in a batch; orders in a buffer are released to the same put wall, and therefore, picked into the same container. Tradeoff: As the number of orders in a batch increases, picking and putting productivity increase, but the cycle time of picking and putting also increase. The cycle time increase will increase variability, which will requires additional accumulation to mitigate.
- Y: Number of put wall batches to be picked simultaneously (assuming a picker can accommodate more than one container on the cart). Tradeoff: As the number of put wall batches increases, picking productivity increases, but picking cycle time increases.
- W: is the number of batches to maintain in the system (the released pool of batches, batches in-picking, batches in transfer to put walls, and at put walls). Tradeoff: As the number of batches increases, throughput increases, but cycle time also increases. As the number of batches increases, the system starts to approach a push system. As the number of batches decreases, throughput is degraded.

Although we are focusing on the pull algorithm here, clearly the underlying design has the greatest impact on the performance of the operation and must be optimized appropriately.

Algorithm: The exact formulas are proprietary, but the general steps for dynamically setting these three parameters are:

For a given setting characterized by:

- Pool of orders in the wave and their associated parameters including number of lines, number of units
- Available number of pickers
- Number of pick zones
- Picking process work content characterized by layout, process delays
- Available put walls
- Cubbies per put wall
- Putting process work content characterized by put wall size, process delays

1. Determine minimum X and Y that meet throughput within capacity constraints (i.e., the current number of workers available).
2. Determine W that maximizes system throughput (i.e., number of orders for a given period of time). In an unconstrained sense, this parameter is influenced by the mean and variance of picking cycle time, putting cycle time, and transportation times from picking zones to individual put walls. Constraints like shipping windows and available dock doors as well as order priorities also have an impact on W.

We run the algorithm every wave to ensure that real-time information can be incorporate into our algorithm to optimize performance over the course of the day. The algorithm, although simply stated, is controlling a dynamic system with many interacting complexities based on the timing over several areas of the DC and has impacts on labor, equipment utilization, and service level. This is why we have invested a great deal of time in developing the algorithm and it has been extensively tested and measured against actual performance.

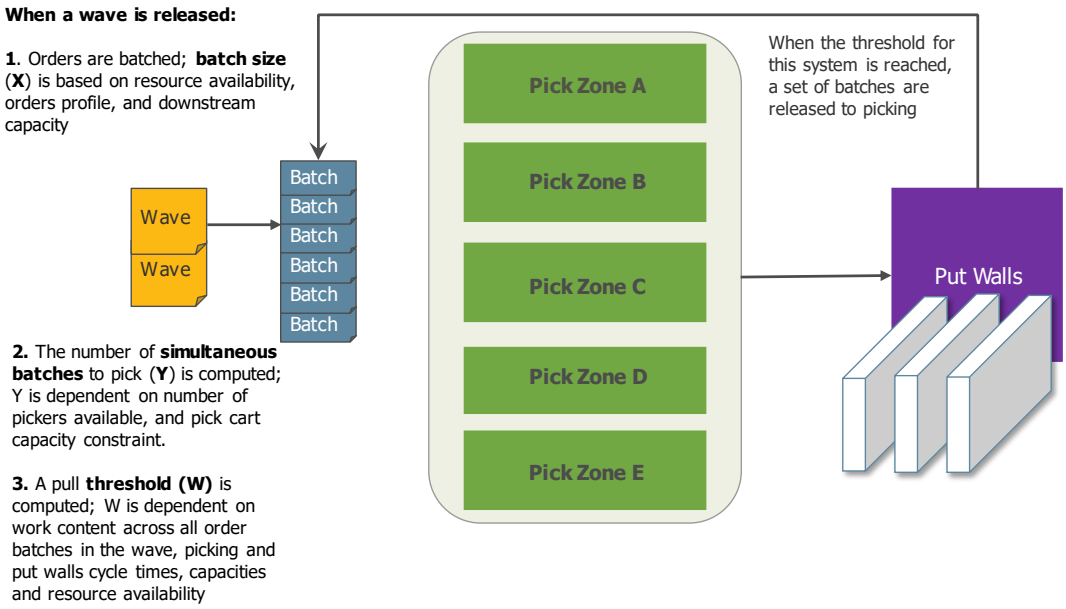

Fig. 4. Pull Framework for Batch Picking to Put Walls

III. Case Study

We were asked to design a distribution center for a major US retailer and determined that a put wall operation was the correct design for fulfilling multi-unit orders for their e-commerce orders. We evaluated the operation under a push- and pull-driven environment to illustrate the value provided by our WES. Cyber Monday is their peak day when they see about 40K multi-unit orders with an average of 4.3 lines/order and 1.2 units/line. SKUs are organized in four pick modules with three levels each, which results in 12 pick zones. The default push approach was to use a batch size of 108 orders per wave and have 50 put walls with about 288 openings (cubbies) per put wall (i.e., the assumed approach for accommodating the variation was to oversize the put walls by 167%).

We applied the heuristic to the operating parameters described above with its assumed setting for headcount and productivity rates in a static environment. The output of the pull heuristic is a batch size of 60 orders on peak day (compared to 108 used for the baseline push system) and put walls with 100 cubbies per wall (compared to 288 in the baseline system). The number of batches to maintain in the system (pull threshold) is 75 batches.

Note that from the design side the pull system has a number of advantages. Smaller put walls (65% smaller), which not only reduce the capital investment tremendously, will have higher productivity as put wall operators travel shorter distances along the wall. However, the smaller batch size will negatively impact picking productivity. Table I provides a number of comparison points and indicates that the productivity pickup at the put walls (43% improvement) outweighs the negative impact on picking productivity (4% reduction). Overall, even ignoring the impact of wave tails (which is greater in the push scenario), there is over a 5% reduction in workers (14 workers over two shifts) using the pull system.

TABLE I. Push vs. Pull Peak Day Settings

	Push	***Pull***
Total Throughput Required	2,000 orders/hour; 8,600 lines/hour; 10,062 units/hour	
Number of Pick Zones	12 zones: FOUR pick modules with THREE levels each	
Number of Put Walls	50	
Put Wall Size (cubbies)	288	100
Batch Size (orders)	108	60
Batches/Pick Cycle	10	6
Picker Productivity	100 Lines/Hour	96 Lines/Hour
Putting Productivity	420 Units/Hour	600 Units/Hour
Release Rule	Release 1000 orders every 30 minutes.	Release a batch when the number of batches in the system go below 75

Note that we chose to hold the total throughput required constant for the two systems and to measure the impact of changes to the time to pick and sort all orders and order cycle time and variation in addition to the investment in the put walls and worker productivity. Other examples can be constructed where, say, the workers are held constant and throughput potential differences are measured. The move to a pull environment provides flexibility over the push environment. Table II summarizes the performance output from the simulation model used to compare the performance of the two approaches.

TABLE II. PUSH VS. PULL PEAK DAY PERFORMANCE

	Push	***Pull***
Hours to pick and sort all orders	24.4	21.7
Average Order Cycle Time (Minutes)	267	103
Order Cycle Time Standard Deviation (Minutes)	19.8	11.2

Figure 5 contrasts the order-by-order cycle time for push vs. pull, and Figure 6 shows the number of lines in process at picking and at the put walls throughout the simulated day.

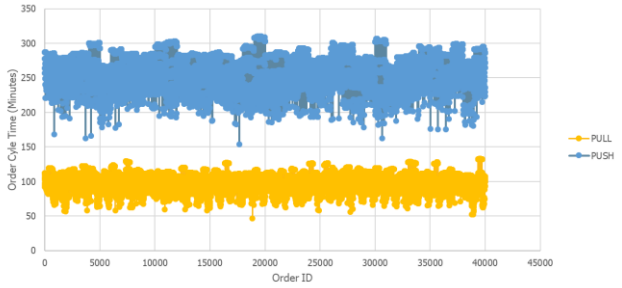

Fig. 5. Order-by-order cycle time in push vs. pull

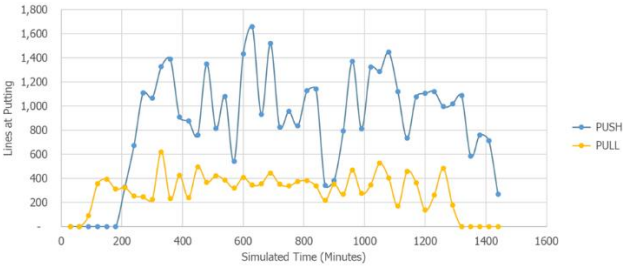

Fig. 6. Variation in number of lines being processed at the put walls over the simulated time. We see a smoother flow in pull vs. push. The smoother flow in pull is more noticeable at the put walls for this case study.

Operationally, not only is the average order cycle time in the pull system 61% lower than in the push system, but we also see much higher variation in the push system cycle time values, which correlates with lower service levels. We also note that in the pull system, the orders were completed 2.7 hours ahead of push.

Under the push framework, operators at the put walls would be idle for extended time in the morning (see figure 6) and when the batches in totes started arriving to the put walls, they would be overwhelmed with work; which would result in an accumulation of totes at the put walls. Operators would then be forced to down-stack the totes on the floor to open space for other totes and unblock the conveyors. This additional work consumes work capacity and reduces productivity.

IV. THE BENEFITS OF PULL

Some of the key benefits of pull-driven vs. push fulfillment that can be realized include:

1. Higher Throughput Capacity: Reclaim the wasted capacity between waves by eliminating the low-productivity transition periods and capacity losses due to queueing and accumulation.

2. Lower Initial Investment: Pull-based control allows smaller batches and reduces the need for large buffers when sizing put walls, which results in dollar and space savings. The downside of smaller batches is the additional number of totes flowing on the conveyor, which could lead to congestion and recirculation. But distribution centers designed for pull operations generally require lower initial investment than those designed to operate with waves because:

- Without the low-productivity wave transitions, facility utilization is higher. The same throughput can be achieved with smaller facilities and less equipment.
- The need for buffers required by wave-based processes is eliminated or greatly reduced.
- In wave-based unit sortation-based operations, most orders seize chutes at the beginning of a wave, but orders do not complete until the tail of the wave. The number of chutes required for incomplete orders peaks mid-wave. Pull processing levels the requirements for chutes by holding incomplete orders in the queue, allowing for designs with fewer chutes.

3. Higher Productivity: In wave-based processes, low productivity periods appear at wave tails and potentially bring operations to a full stop. In pull processes, stockouts and other unexpected events affect only the orders they belong to and all other resources can continue working without any delay. Picking productivity is higher with pull processing because it eliminates work starvation periods for the pickers created by wave transitions. And pull processing reduces pickers' travel time. Pull processing takes a system wide view to ensure that workers with the same capability are working consistently by ensuring that their output is ready to be received downstream.

4. Better Handling of Rush Orders: In wave-based processes, emergency orders are often held to be assigned to an upcoming wave where they will have a minimal impact on productivity. With pull-driven processing, the emergency order can be inserted as the next released order (or as the highest priority order to process) without any impact on the productivity of the operation.

5. Enhanced Customer Service: Typically, we see between 20% to 60% reduction in average order cycle times. The real-time nature of pull-driven processing allows the distribution center to better manage shipping deadlines. If a distribution center is processing 50 orders and realizes that the next 30 orders in line are at risk of missing their deadline, a hold can be placed on the other orders to speed up the processing of the currently at-risk orders. This hold can be cancelled when the situation is rectified. Such an approach is very difficult to process in a wave-based system.

V. CONCULSIONS AND FUTURE WORK

We conclude three things from this research. First, a pull-based flow for fulfilling orders can vastly improve the operational performance including reducing the cycle time,

meeting throughput requirements, and leveling resource utilization. Second, the interdependency between batch sizes, human resource availability, work content, and productivity need to be modeled and understood well prior to setting the pull parameters. There are unintended consequences for smaller batches, such as the need for more people or more containers and more carts to keep the work flowing, or resource starvation, which makes DC manager and operators very nervous. Therefore, a flexible workforce that can react quickly to shifting resource allocation needs, and a thorough system-wide tradeoff analysis needs to be conducted prior to an architectural software design and post implementation to calibrate and revise parameters. Third, folks in industry are often skeptical of the concepts we presented here and the belief that larger batches and continuously pushing the work is the preferred method of operation. Much work has gone into making the case and demonstrating the unintuitive effects of variability and lost capacity.

Next, we are planning to expand the pull framework to include multiple fulfillment channels when inventory and resources are shared across these channels.

REFERENCES

[1] M. L. Spearman, W. J. Hopp, and D. L.Woodruff, "A hierarchical control architecture for Constant Work-in-Process (CONWIP) production systems," in Journal of Manufacturing and Operations Management, vol 2, pp. 147–171, 1989.

[2] J. Prakash, and J. F. Chin, "Modified CONWIP systems: a review and classification," in Production Planning and Control, vol. 26: 4, pp. 296-307, 2015.