

Addressing Retail Out-of-Stock Issues Using System Dynamics

Sheng Liu
VP, Business Intelligence
CROSSMARK, Inc.
Plano, TX
www.crossmark.com

J. Chris White
President
ViaSim Solutions
Rockwall, TX
www.viasimsolutions.com

Abstract:

“Out of Stock” (OOS) has long been a plaguing problem in the consumer packaged goods (CPG) industry for both manufacturers and retailers. It refers to a situation in which an item is unavailable for sale as intended at a store. OOS not only leads to lost sales in the short term, but also may lose customers in the long run. One study estimated that OOS items on average cost retailers 4 percent of their annual sales, and manufacturers \$23 million for every \$1 billion in sales. Both manufacturers and retailers are motivated to understand how and why OOS occurs so that it can be prevented, or at least minimized. OOS could be caused by many factors, such as manufacturer production shortage, distribution center delay, consumer demand surge, and sub-optimal store operations, etc. There have been many previous attempts to model and fix OOS. To the authors’ knowledge, this is the first study to address the full production-distribution system using system dynamics (SD) to help all players in the system understand the dynamics of the OOS event, the structure of the CPG distribution system that contributes to the system behavior that causes OOS to occur, and interventions or improvements that can be made to the CPG distribution system to improve its performance related to OOS events. This paper will describe the OOS model that was developed, the insights that were gained, and the process used to translate the initial SD model into a business application that can be employed on a wider range of product and retailer scenarios.

Introduction

Out of stock (OOS), a retail problem where an item runs out on shelf and/or in store, has long been a plaguing problem for both manufacturers and retailers. It is well recognized that the adverse impact of OOS much exceeds the lost sales of the OOS item alone (Gruen and Corsten 2007, Anderson et al 2006). When consumers encounter OOS for an item on their shopping list, their two most likely reactions in the short term are (Gruen and Corsten 2007): to switch to a different brand, in which case the manufacturer loses sales, and also gives its brand competitor a valuable trial opportunity; or to buy the item at another store, in which case the retailer loses sales to a retail competitor. In the long run, consumers may permanently adjust their shopping activities, including both choices of store and item. In addition, OOS distorts demand signal and as a result future supply planning may be underestimated, which further reinforces the OOS situation.

OOS is an even more important problem to address in today's environment where credit is tight and spending is cautious. On the one hand, it is very costly for manufacturers and retailers to carry excess inventory that does not sell quickly. On the other hand, OOS costs them the sales they have earned through expensive marketing programs in competition for consumers' tightly held wallets. Furthermore, continued SKU proliferation has led to increasingly crowded shelf spaces. Intuitively, the fewer spaces an item has on shelf, the more likely it will be OOS.

In this paper, we focus on OOS caused by sub-optimal store operations, which is responsible for between two-thirds and three-fourths of OOS occurrences (Corsten and Gruen 2003). We use system dynamics model to study the interactions of various factors in play at store level that may lead to OOS. We hypothesize that shelf space allocation and re-stocking practices are the two most critical store operation decisions. We expect our findings could be used to minimize OOS through dynamically guiding efficient inventory management, shelf space allocation, and store personnel deployment.

The Out-of-Stock (OOS) Model

Figure 1 shows a pictorial diagram for a general distribution process where OOS may occur. Products travel from a warehouse and are delivered to a retail store. At the retail store, products are moved from a receiving inventory onto the "floor" of the store to be made available for purchase by consumers. Based on current inventory, the retail store may order additional products to be delivered from the warehouse. There is much more to the entire production-distribution-sales process. However, as we choose to focus on OOS caused by sub-optimal store operations, we will only look at the flow of products from the warehouse to the consumer. Even in this truncated view of the full process, we are mostly concerned with what happens in the retail store to see how the process structure drives OOS behaviors.

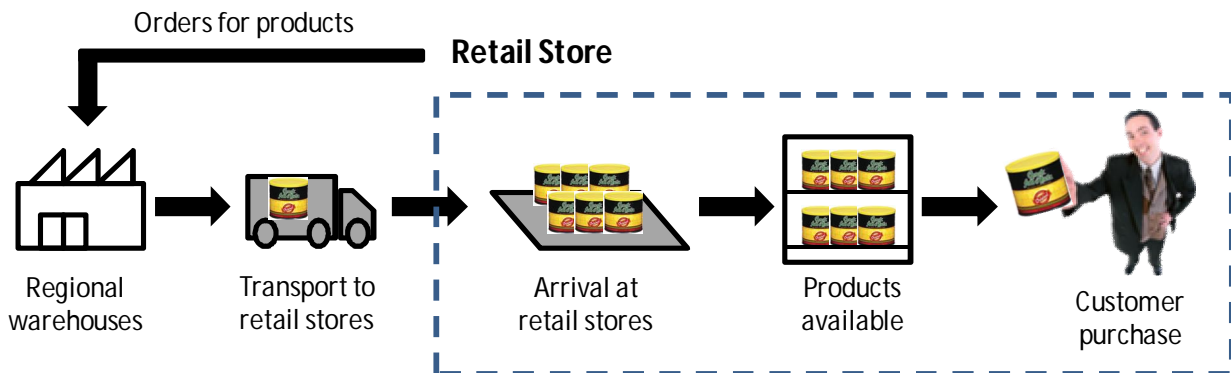


Figure 1: Overview of a general distribution process where OOS may occur

For the portion of the total system shown in Figure 1, a simple causal loop diagram (CLD) for the OOS process is shown in Figure 2. Essentially, this is a basic inventory correction feedback loop that operates to keep a desired level of inventory in the store (available for consumers to purchase).

Addressing Retail Out-of-Stock Issues Using System Dynamics

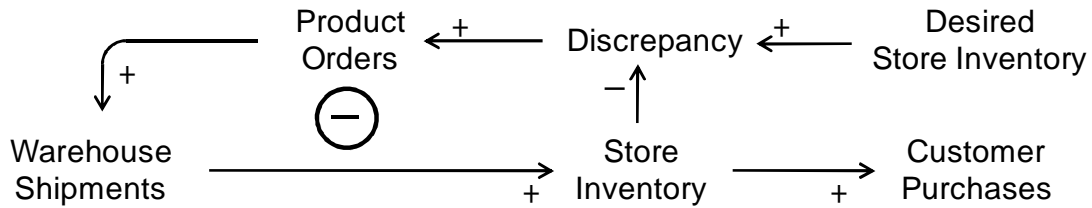


Figure 2: Basic Causal Loop Diagram (CLD) for OOS Process

Figure 3 shows a simplified version of the resulting system dynamics OOS model. Several variables have been removed from the diagram to help with visibility. For example, *Shipping*, *Receiving*, *Picking*, and *Facing* are all conducted on specific schedules so they “fire” (discretely) every once in a while and are not continuous. The variables with these schedules have been removed to reduce visual clutter. In addition, portions of the model related to bookkeeping and ordering additional products from the upstream warehouse are not included in this diagram because they are ancillary to the product flow. Finally, the operations of the warehouse are removed because the warehouse is external to the retail store, which is the main focus of Figure 3.

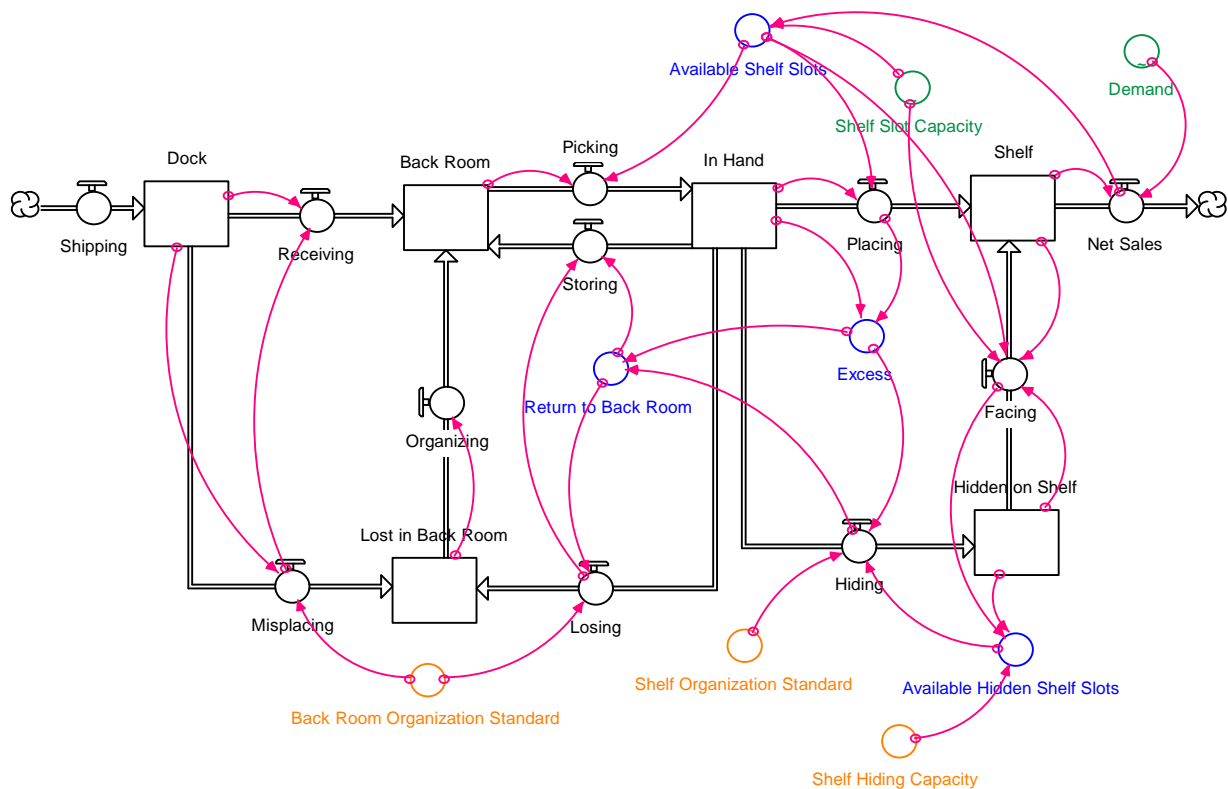


Figure 3: Overview of OOS model

Moving left to right in the diagram, products are shipped from a warehouse (which may be a central warehouse for the retailer or a distributor that services multiple retailers) and arrive on the *Dock* of a retail store. In this sense, *Dock* represents any material receiving component of the retailer. The *Dock* could be a true physical area outside the retail store where products arrive on pallets and await

movement into the store. Or, depending on the size of the retail store, the *Dock* could be a virtual representation of arrival of products to a place inside the store where products wait to be moved to the floor. This storage area within the store is designated as the *Back Room* and represents all storage of excess products that are not on a *Shelf* (i.e., publicly available for purchase by customers). In either example of delivery, the key is that products eventually arrive in the *Back Room* of the store through *Receiving* and these products are to be moved at a later time onto the floor (i.e., *Shelf*).

Based on how many "slots" are open on the *Shelf* for the product (referred to as *Available Shelf Slots* in Figure 3, which is a function of *Shelf Slot Capacity*), product units are "picked" from the *Back Room* (through the flow called *Picking*) and carried or transported onto the "floor" of the store for placement. When products are being moved from the *Back Room* to the *Shelf*, they are considered *In Hand*. These products are then placed on the *Shelf* (through the flow called *Placing*) to fill the available open slots. Once on the *Shelf*, these products are available for customers to purchase. In this model, the *Demand* variable in the upper right corner of the diagram represents customer demand for the specific product.

While carrying or transporting products to the floor (i.e., *In Hand*), occasionally there are extra products that do not fit within the space available on the *Shelf* for the specific product. In Figure 3, this is designated as *Excess*. This *Excess* can be due to incorrect inventory counts that show fewer products on the *Shelf*, or it can be due to the fact that products are shipped in "cases." For example, a can of dog food may be shipped in cases of 12 cans (units). Rarely is a case opened in the *Back Room* and the precise number of needed cans taken out. Instead, a case is taken to the floor and opened there. The appropriate amount of cans are removed and stocked on the *Shelf* (e.g., 5), and the remaining cans in the case (e.g., 7) are taken back to the *Back Room*. This process of returning *Excess* products to the *Back Room* is represented by the flow called *Storing*.

However, notice that there are other options for doing something with *Excess* products. The person stocking the products on the *Shelf* may not have room to put additional products in the specified placement area for that particular product, but there is ample room somewhere else on the *Shelf* (where a different product is supposed to reside). This process is called *Hiding* products on the *Shelf* and results in a stock of products that are *Hidden on Shelf*. Periodically, these hidden products are found during shelf organization processes. This organization process is called *Facing* the product and involves moving the products to their proper slots so they are easily found by customers. In the model, *Shelf Organization Standard* represents how well a retailer keeps its shelves organized. If the *Shelf Organization Standard* is low (close to 0), the retailer is rather sloppy with its placement of products, and products are routinely stuffed on shelves in the wrong places. Conversely, if the *Shelf Organization Standard* is high (close to 1), the retailer is very organized and only places products in their designated spaces.

As another option, sometimes when products are *In Hand* and being returned to the *Back Room*, these products are stored in the wrong location (through the flow called *Losing*) and end up *Lost in Back Room*.

This idea of misplacing things in the *Back Room* requires a few other variables in Figure 3 to be highlighted. When products arrive on a *Dock* and later be moved into the *Back Room*, there is a chance that the retailer personnel can misplace the products. This means that products are stored in the *Back Room* in a location that is not designated for that specific product. This is represented by the *Misplacing* flow in the lower left of the diagram. When product is misplaced, it is considered *Lost in Back Room* (lower left stock in Figure 3). The products are actually in the *Back Room*, but not where they are

supposed to be stored. Periodically, the *Back Room* is “organized” and lost products are found and moved to their correct storage spots. This clean-up process is represented by the flow *Organizing* that moves products up from *Lost in Back Room* into *Back Room* (where they can now be pulled and moved to a *Shelf*). In the model, *Back Room Organization Standard* represents how well a retailer keeps its *Back Room* organized. If the *Back Room Organization Standard* is low (close to 0), the retailer is unorganized and routinely misplaces or loses products in the *Back Room*. Conversely, if the *Back Room Organization Standard* is high (close to 1), the retailer is very organized and only places products in their designated spaces in the *Back Room*.

Overall, the general flow that typically happens is that products are received at the *Dock* of the store, scanned into the inventory system, and moved into the *Back Room*. Based on whether or not there is available space on the *Shelf* for the product, products are taken *In Hand* and moved to the *Shelf*. Since products typically come in cases, the cases are moved to the floor for stocking on the *Shelf*. Extra products that do not fit in the available space on the *Shelf* are then returned to the *Back Room*.

Most retailers have regular *Picking & Placing*, or re-stocking schedules, such as every night, every Friday night, etc. Each re-stocking operation incurs labor cost and is usually done for at least an entire category of items. In other words, there is a tradeoff in store operations between the cost of re-stocking the shelf and the benefits of a fully-stocked shelf without OOS.

Manufacturers often hire third party service providers, such as CROSSMARK, to address their OOS problems. Third party service providers perform re-stocking services for their clients' products in addition to a store's regular operations. This supplemental re-stocking usually is in the form of continuity coverage. For example, CROSSMARK re-stocks one of its client's products every weekend when OOS is most likely to happen during those shopping peak times while regular store operations could not maintain stocks on shelf. A supplemental re-stocking is also often done in response to a brand promotion or higher expected sales rate. Some third party service providers, such as CROSSMARK, also have relationships with retailers that allow them to suggest or place a store ordering when encountering a store OOS or find that store inventory level is too low.

NOTE: For the purpose of this paper, several details are not discussed to keep the content simple and digestible within the scope of a short paper. However, the description above for Figure 3 is enough to gain an understanding of the major dynamics for the OOS issue.

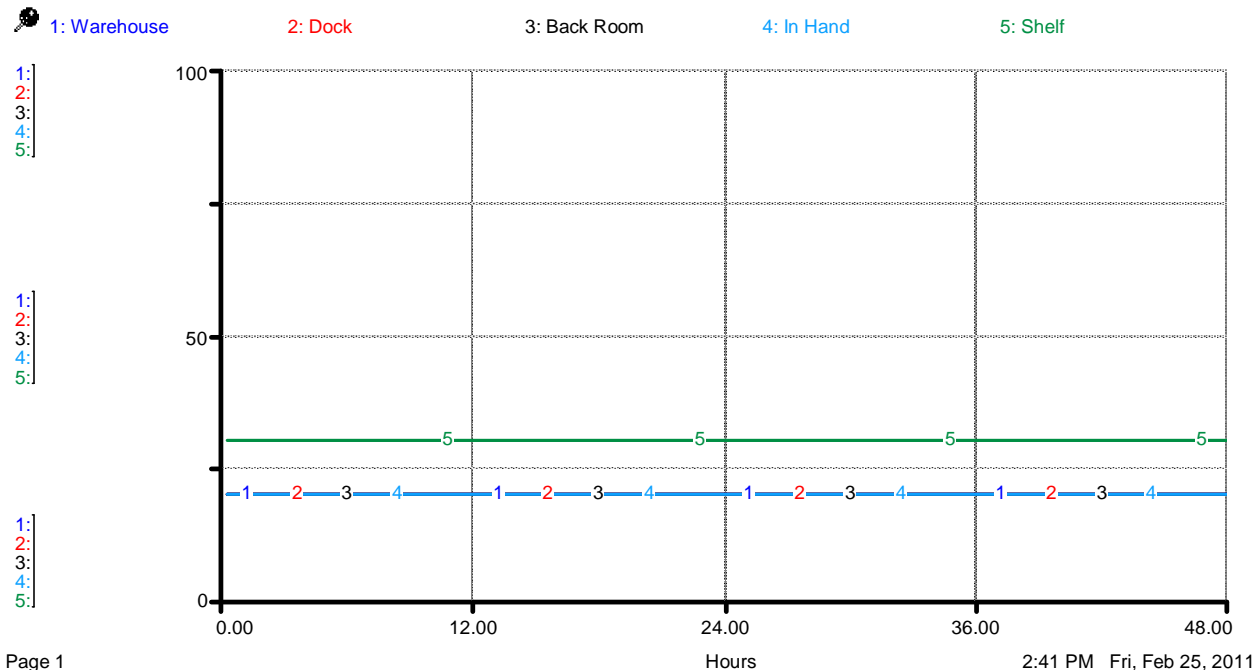
Validation of the OOS Model

The graphs below will show some of the output results from the OOS model in iThink. The purpose of showing these results is to give the reader a sense of the dynamics that occur in the process. For all graphs except Figure 8, the five key stocks are shown: *Warehouse, Dock, Back Room, In Hand, Shelf*.

Figure 4 shows results from the baseline model for a 48-hour simulation run. In this model, demand is constant at 20 units per hour. Ordering from the warehouse is also constant at 20 units/hour. Of course, in the real world, orders and deliveries would not occur on an hourly basis. And, a retail store would not see the same level of demand for all 24 hours of operation (assuming it is a 24-hour store). However, this case is used to ensure that the model initializes in equilibrium and behavior is correct for continuous flow. If continuous flow is not correctly tested, it can be difficult to test a model that is driven by periodic ordering. It can be unclear whether test issues arise because of the “discrete” nature of the periodic ordering or because of the fundamental underlying structure. By running the model with

Addressing Retail Out-of-Stock Issues Using System Dynamics

a continuous flow, we can isolate any test issues that arise from the underlying structure. For the purpose of examining output results, the continuous example of a constant flow of 20 units/hour will be used. In this baseline case, we should see a level flow of 20 units/hour through each stock in the model (i.e., *Warehouse, Dock, Back Room, In Hand*), as seen in Figure 4. The only stock that is different is the *Shelf* because it has space to hold 30 units.



?

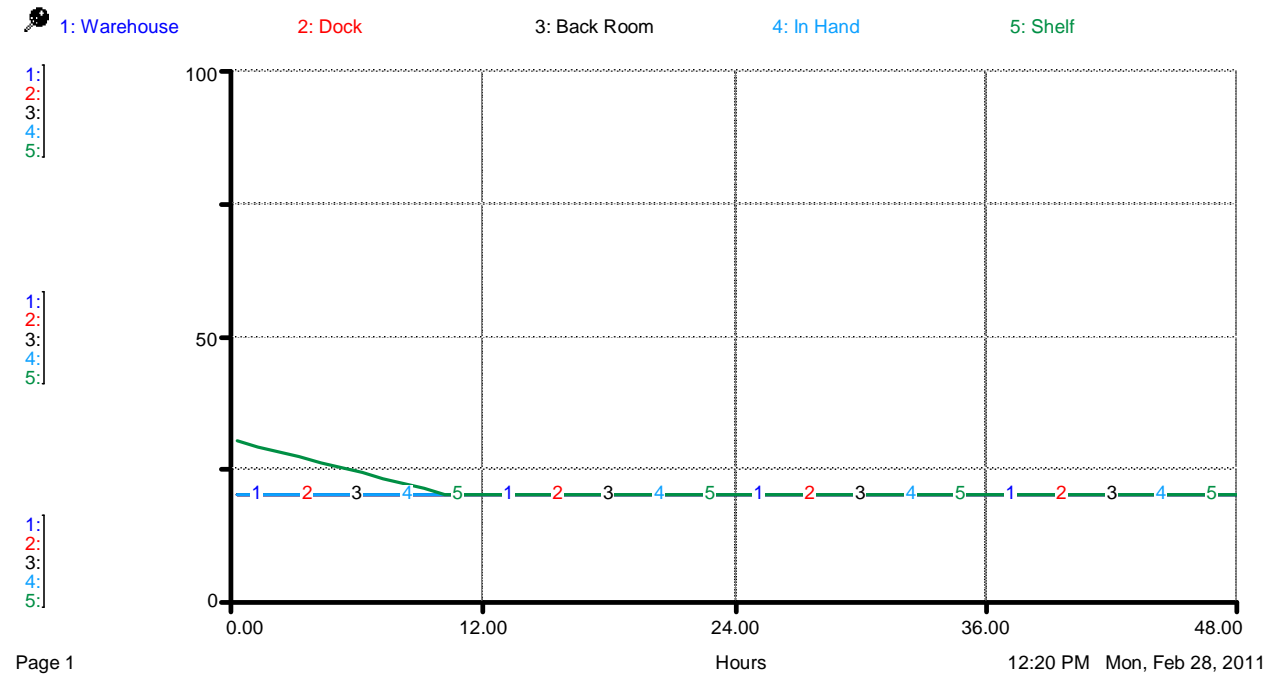
Figure 4: Baseline Results: Demand = 20

For the first test scenario, customer demand was increased to 21 units/hour. With an order rate of 20 units/hour and a demand of 21 units/hour, we should see any excess units available get “pulled” from the system at a rate of 1 unit/hour until an equilibrium is met at the constant level of ordering (which is 20 units/hour). In Figure 5 this is exactly what we see. The Shelf begins with 30 units. It loses 1 unit/hour for the first 10 hours of the simulation until the Shelf stock reaches 20 units (the same level as the order rate). All other demand will be left unfulfilled. Obviously, this is not the situation the retailer and the manufacturer want because it signifies lost sales. However, for the purposes of model testing, this is the correct behavior for this set of conditions.

For the next test scenario, customer demand was reduced to 19 units/hour. With an order rate of 20 units/hour, the expected behavior is that the additional 1 unit/hour that is being “pushed” into the system (but not “pulled” because of lower customer demand) will accumulate somewhere in the system. If the *Back Room* organization standard is high, then no items will be *Lost in Back Room* and we should see an accumulation of inventory in the *Back Room*. Because the order rate exceeds demand, we should see the number of units on the Shelf stay at 30 units. We should also see that the number of units that are moved from the *Back Room* to the *Shelf* through the *In Hand* step of the process matches the demand level of 19 units/hour. In Figure 6, we see all three of these expected results.

Addressing Retail Out-of-Stock Issues Using System Dynamics

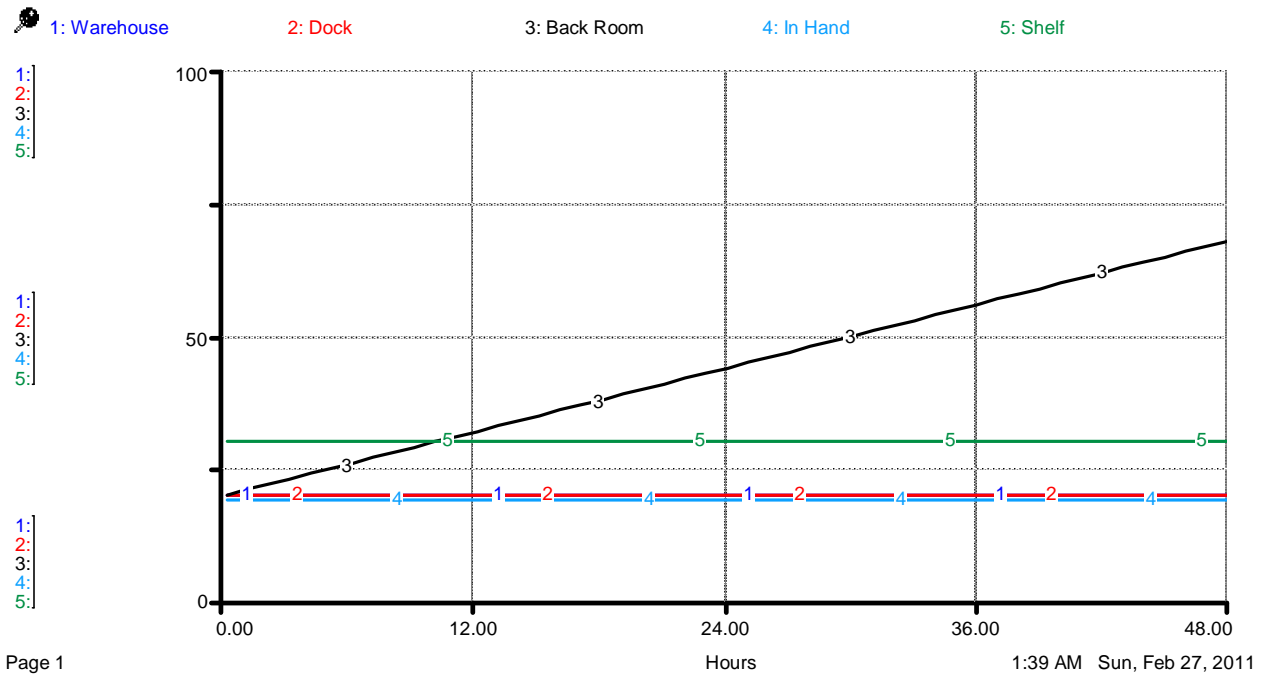
The next dynamic we want to test is whether or not the model properly allows hiding products on the shelf (*Hidden on Shelf*). This test is an extension of the previous test shown in Figure 6. For this new test, we should see the first few excess units move to *Hidden on Shelf*, based on the number of *Available Hidden Shelf Slots*. If *Available Hidden Shelf Slots* = 10, we should see the first 10 excess units (in the first 10 hours of the simulation) go to *Hidden on Shelf* instead of the *Back Room*. Once the number of products *Hidden on Shelf* reaches the maximum *Available Hidden Shelf Slots*, we should see excess products accumulate in the *Back Room* (starting at Hour 11 of the simulation). Figure 7 shows this correct behavior. For the first 10 hours of the simulation, there is no accumulation in the *Back Room*. In Figure 8, it is verified that the number of items *Hidden on Shelf* reaches and stays at 10.



?

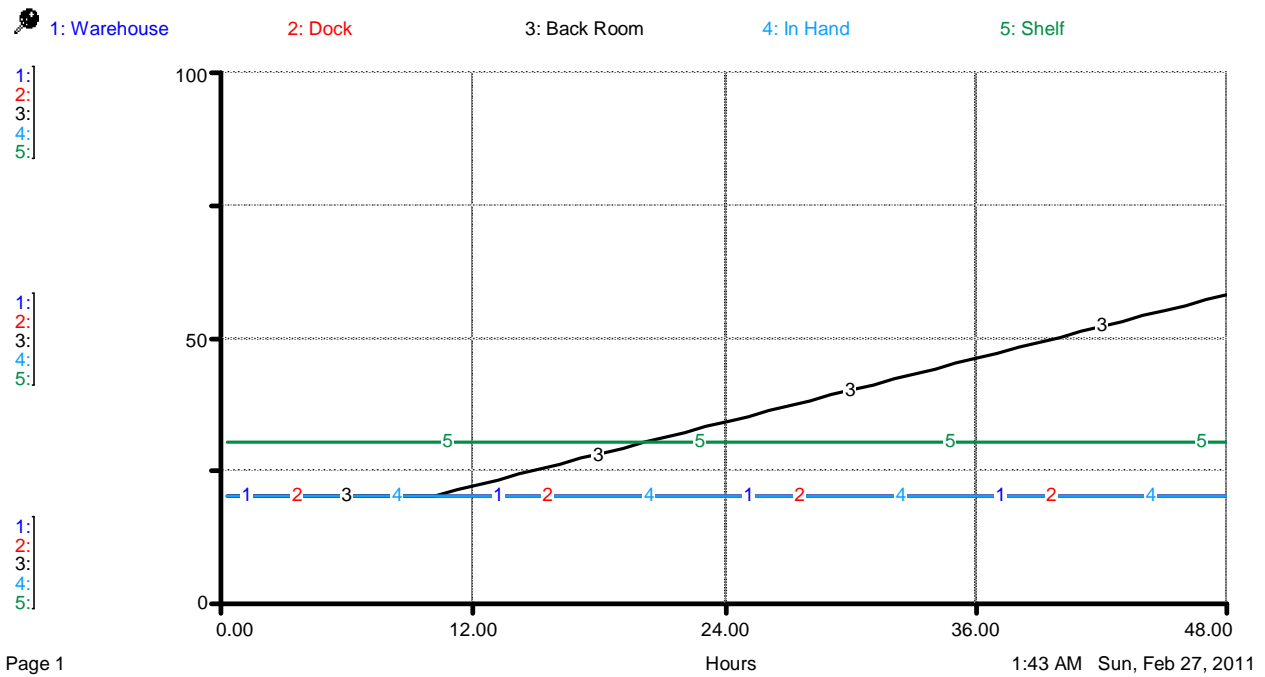
Figure 5: Test Results: Ordering = 20, Demand = 21

Addressing Retail Out-of-Stock Issues Using System Dynamics



?

Figure 6: Test Results: Ordering = 20, Demand = 19



?

Figure 7: Test Results: Ordering = 20, Demand = 19, Available Hidden Shelf Slots = 10

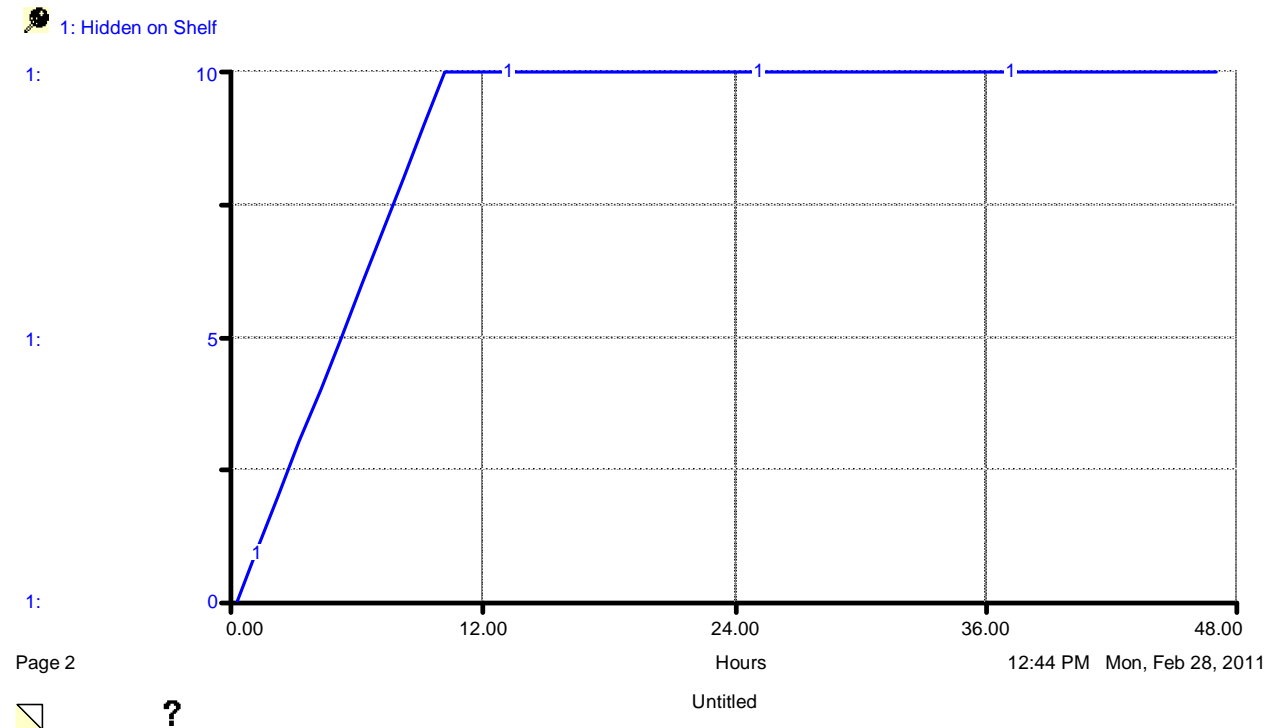


Figure 8: Test Results: Ordering = 20, Demand = 19, Available Hidden Shelf Slots = 10

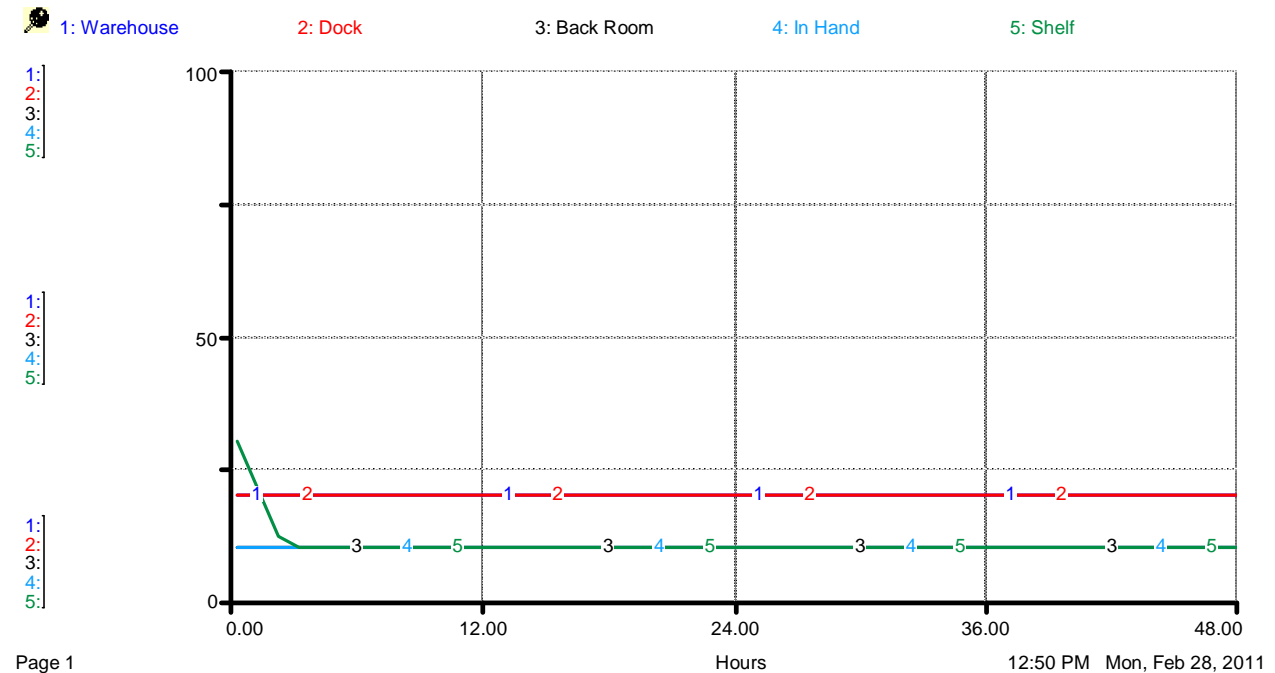
For the next scenario, we removed the ability to hide items on the shelf and focused on losing items in the *Back Room* (i.e., *Lost in Back Room*). If the *Back Room Organization Standard* is set to 0.50, this indicates that half of the items moved from the *Dock* into the *Back Room* are *Lost in Back Room*. For this test, we should see only half (i.e., 10) of the ordered units moving through the retailer stocks. In essence, this is similar to the scenario in Figure 5 in which demand was greater than the order rate because demand is 19 units/hour but the flow of units to the *Shelf* should only be 10 units/hour. We should see behavior similar to that shown in Figure 5, except that the excess available items on the *Shelf* will be consumed much more quickly (instead of 1 unit/hour it is now 9 units/hour) and the final steady flow should be 10 units/hour through the *Back Room*, *In Hand*, and *Shelf* stock levels. Figure 9 shows this behavior. In addition, we should see the amount of units *Lost in Back Room* increase at the rate of 10 units/hour. This is shown in Figure 10.

One final key behavior to test was the situation in which items are *Lost in Back Room* and the *Back Room* gets organized. In this case, we should see units move from *Lost in Back Room* to the *Back Room* (so they are now available to be picked and put on the floor for sale) whenever this organizing action takes place. Figure 11 shows the results of the previous scenario (ordering = 20, demand = 19, 50% back room organization) with the addition that every 4 hours the *Back Room* is organized (through the flow *Organizing*) and items are found and moved to their correct locations. Of course, in the real world, this type of organization activity (if it is needed) occurs less frequently. But, we are just testing behavior, so setting the 4 hour interval is sufficient and it is easy to validate. In Figure 11, the simulation starts off the same as shown in Figure 9 for the first 4 hours. Then, when the *Back Room* is organized and items that are *Lost in Back Room* get found and moved, we see the inventory level for the *Back Room* spike up as 40 items are moved from *Lost in Back Room* to the *Back Room* (because these items were accumulating at a rate of 10 units/hour for 4 hours). Once at the elevated level of inventory, the *Back Room* now has enough inventory to satisfy demand (19), so we do not see the *Back Room* inventory get

Addressing Retail Out-of-Stock Issues Using System Dynamics

down to the previous low level of 10 units. Instead, as with Figures 6 and 7, we see the *Back Room* inventory rise jaggedly as groups of items are periodically moved from *Lost in Back Room* into *Back Room*.

Now that we know behavior is consistent and logical in the OOS model, we can begin to experiment with the model to gain insights (which are often counter-intuitive at first glance) into overall system behavior.

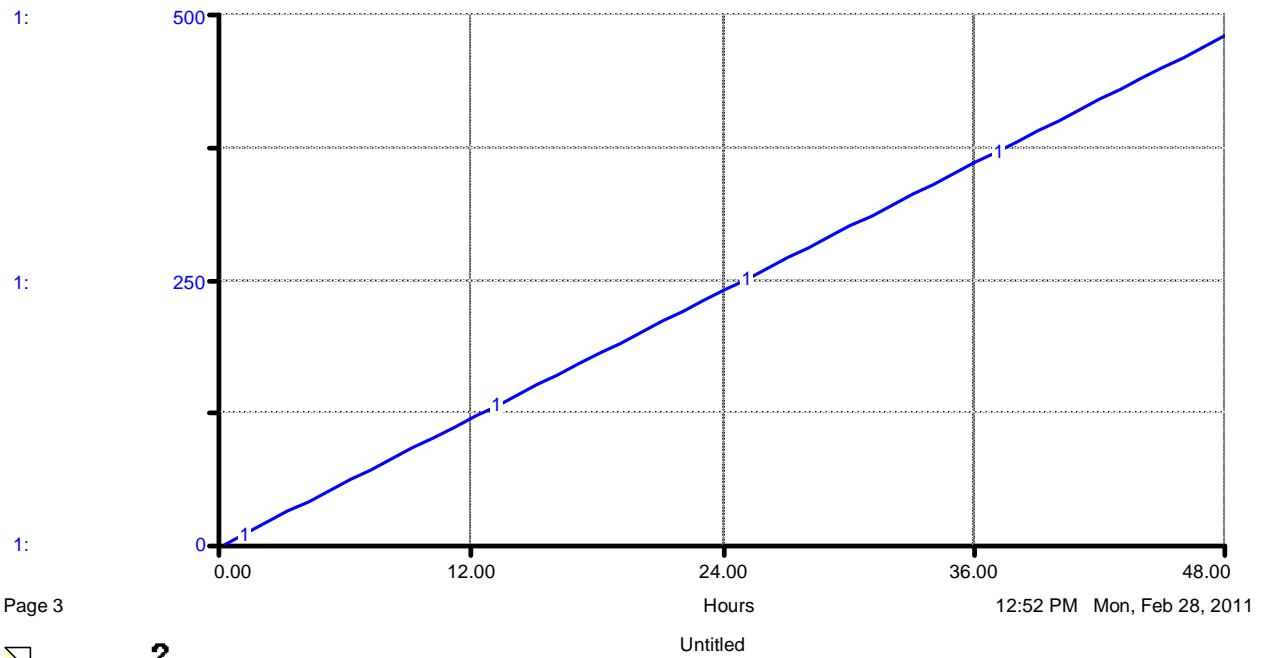


?

Figure 9: Test Results: Ordering = 20, Demand = 19, 50% Lost in Back Room

Addressing Retail Out-of-Stock Issues Using System Dynamics

1: Lost in Back Room



Page 3

?

Figure 10: Test Results: Ordering = 20, Demand = 19, 50% Lost in Back Room

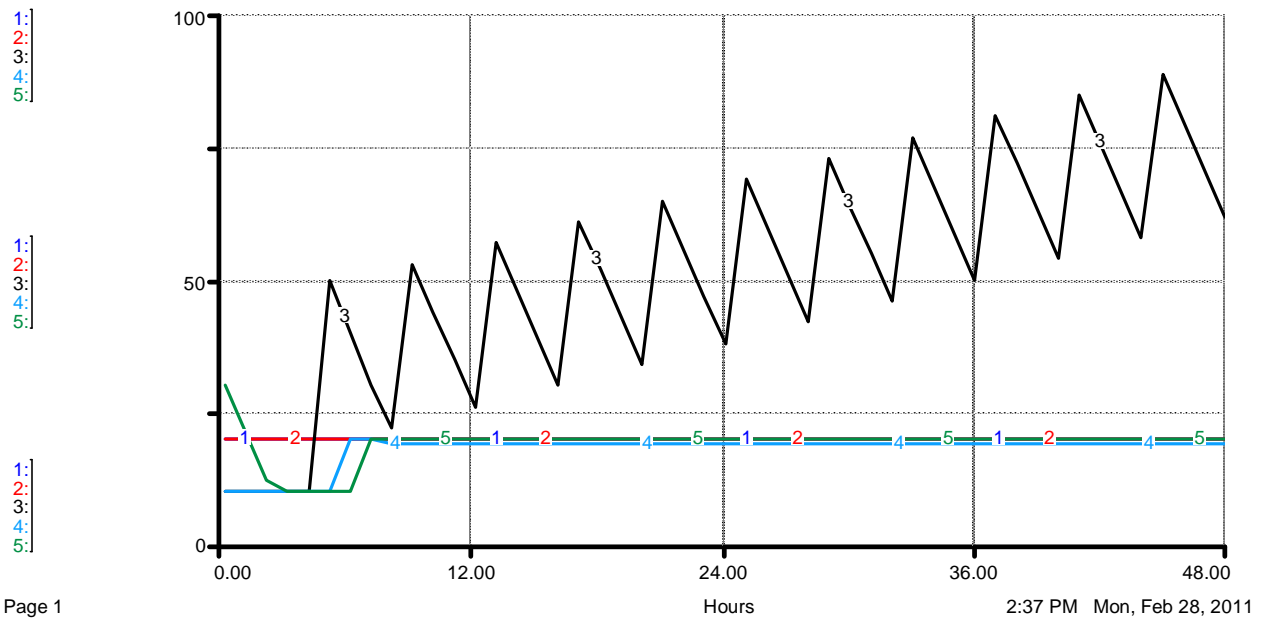
1: Warehouse

2: Dock

3: Back Room

4: In Hand

5: Shelf



Page 1

?

Figure 11: Test Results: Ordering = 20, Demand = 19, 50% Lost in Back Room, Organizing each 4 hours

Insights Gained from the OOS Model

The OOS model provides several insights that, in many ways, demonstrate the characteristic role of SD for identifying counter-intuitive behavior and highlighting that we (ultimately, through the system structure) often cause our own problems.

Let us begin with the front-end of the entire process: customer demand. Customer demand is strongly driven by customer satisfaction. If customers are happy with a product and/or buying experience, they will buy again, as well as “spread the word” to their friends and family. One way that manufacturers and retailers use to increase customer satisfaction is to increase the product offerings to make sure every segment of the customer population has a product that meets their needs. Because stores have limited physical space available on the floor to display products, an increase in the number of products placed on any given shelf decreases the available space allocated to any one product (on average). If a retailer has 10 feet of shelf space with 10 products, each product gets an average of 1 foot of shelf space. If the number of products doubles to 20, then each product gets an average of 6 inches of shelf space. When there is less shelf space, more products would need to be stored in the back room. Because the shelves are emptied more quickly (assuming the same level of customer demand for a product but less available products on the shelf due to smaller shelf slots), re-stocking would have to be more frequent (compared to the previous situation with larger shelf slots) in order to prevent OOS and capture all the sales.

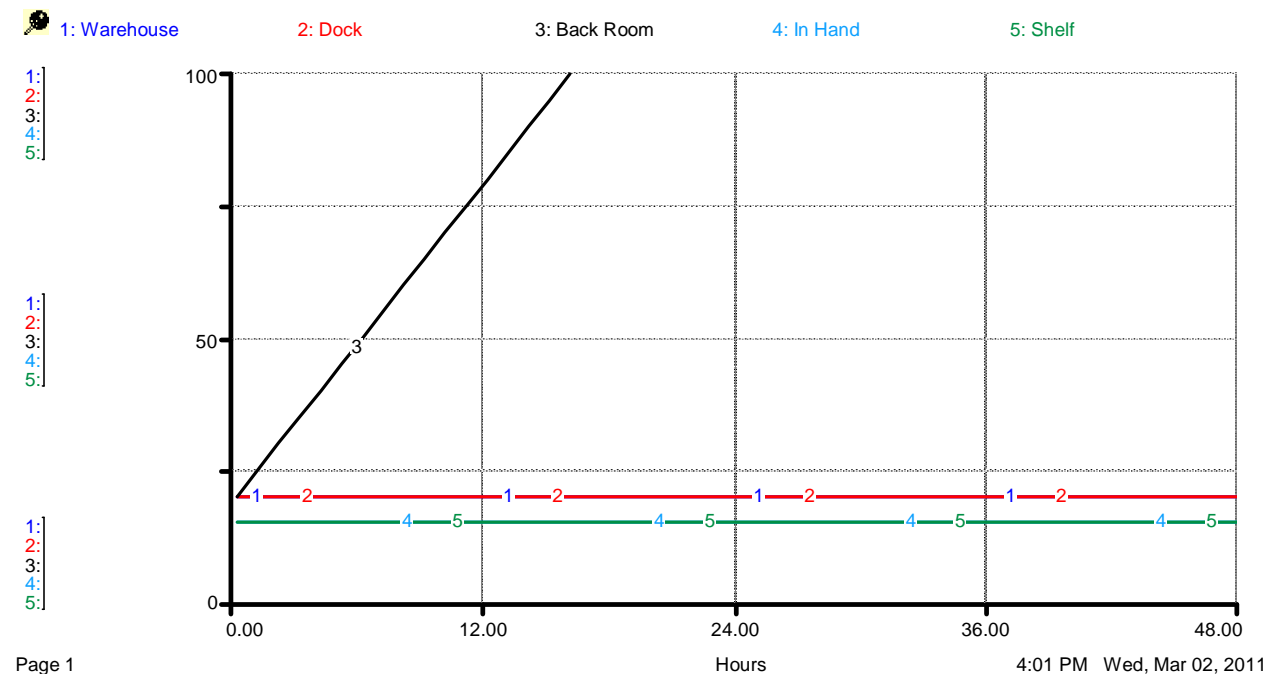


Figure 12: Demand = 20, Shelf Slot Capacity = 15

Figures 12 and 13 illustrate the consequences of this scenario if re-stocking schedule is not made more frequent. Starting from the baseline case as in Figure 4, instead of having space for 30 units on the Shelf, the slots are reduced to 15 spaces. Demand and ordering are still set at 20 units/hour. In Figure 12, notice that the inventory in the Back Room increases by 5 units each hour because restocking remains at only every hour and, thus, is limited to the Shelf Slot Capacity (15). Since demand is 20

units/hour, the store misses 5 sales per hour. This result can be seen in Figure 13, which shows the total missed sales. Notice that it rises at the rate of 5 units/hour, representing a constant state of OOS.

Given the limitation of *Available Shelf Slots*, the only way to prevent OOS would be to increase the frequency with which restocking occurs. If restocking occurs every half-hour (and assuming demand is fairly constant throughout each hour), then the system would be able to support the demand of up to 15 units/half-hour with a shelf capacity of 15 slots with no OOS.

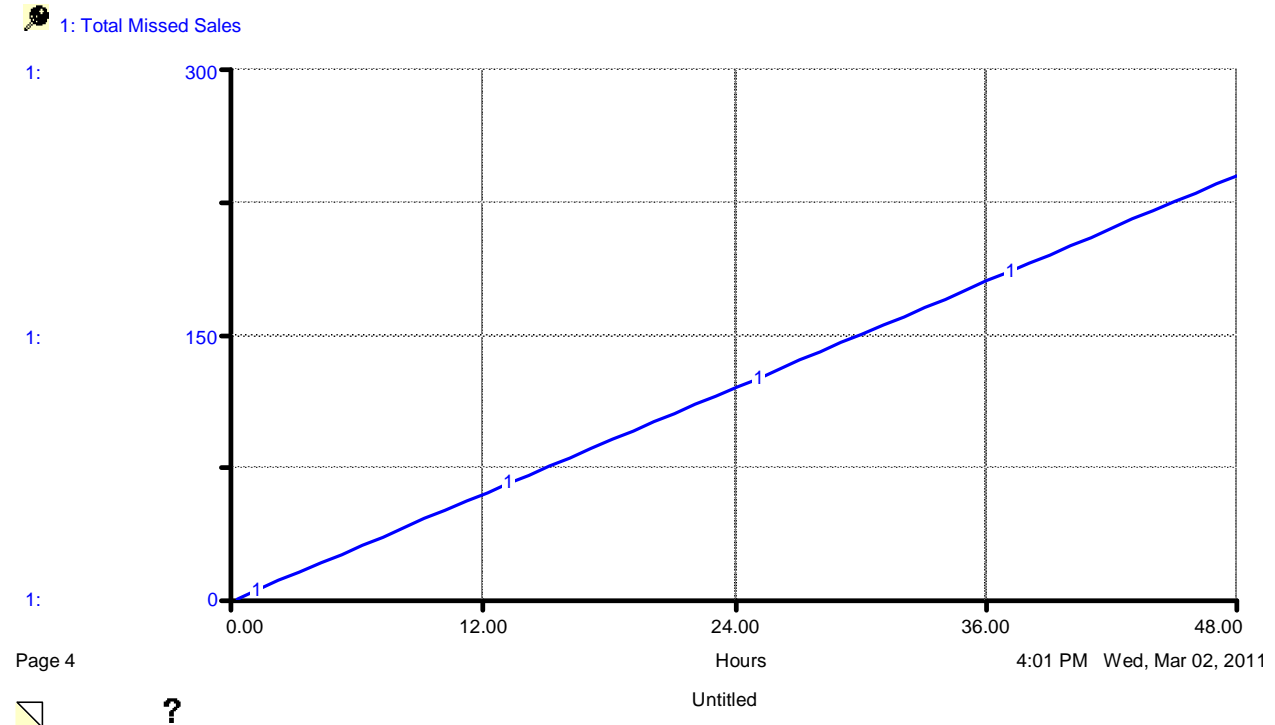


Figure 13: Demand = 20, Shelf Slot Capacity = 15

This illustrates a three-way trade-off among product availability, inventory space, and labor costs (Figure 14). As with similar trade-offs, you can achieve two of the objectives but at the expense of the third objective. For example, as pointed out above, a store can achieve high product availability (no OOS) with a given inventory (*Available Shelf Slots*) at the expense of higher labor costs (more frequent restocking). Or, conversely, a store can save labor costs with the same inventory by limiting product availability (i.e. tolerate high OOS). If the store wants high product availability and low labor costs, more inventory space would be required (more shelf space).

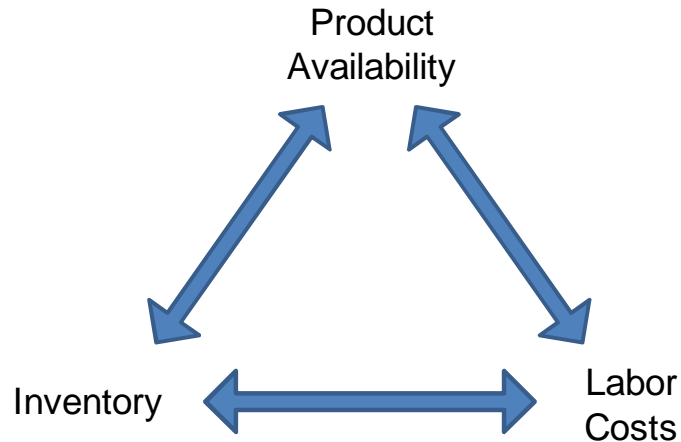


Figure 14: Three-Way Trade-Off Among Product Availability, Inventory, and Labor Costs

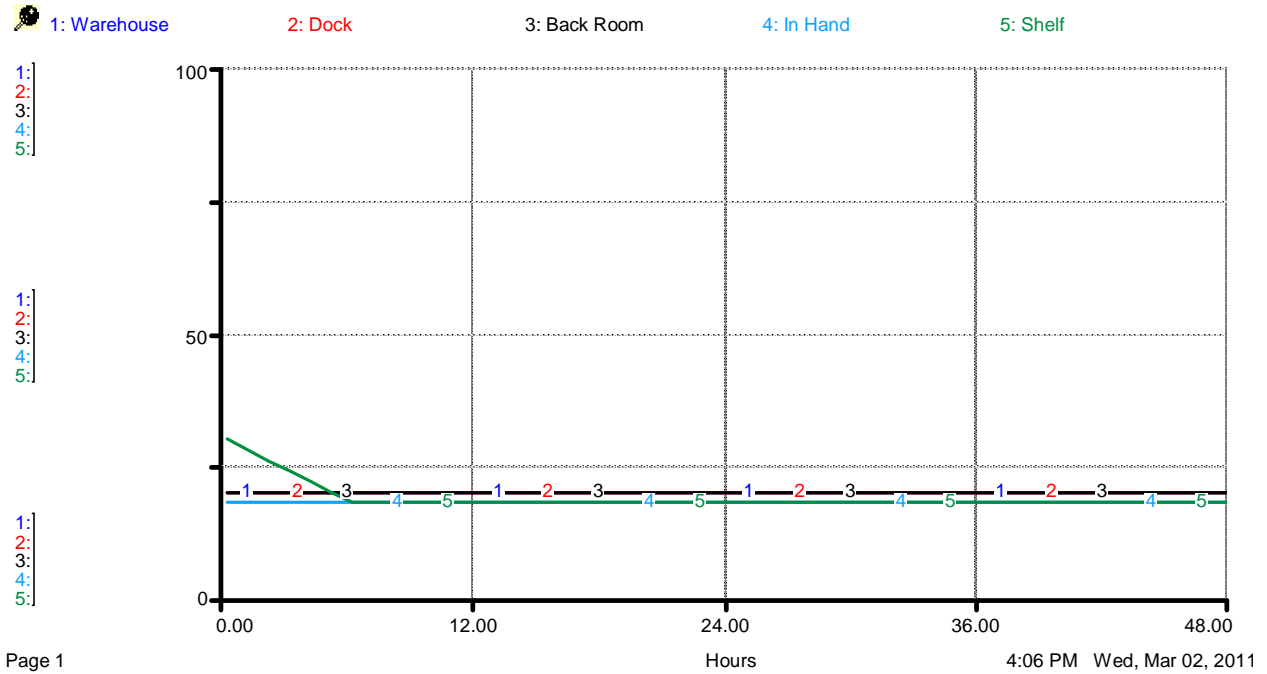
Next, let us examine a retail problem called “shrink”. Shrink refers to the loss of products due to damage, theft, or permanent misplacement (i.e., will never be found). In other words, these products are gone and no longer available. [NOTE: In some situations damaged products are not considered shrink because they can be sent back to the manufacturer through a reclamation process.] High shrink rates is a major cause of OOS (and cost) because products that are ordered and shipped to the store do not make it to the consumer.

As more products are stored in the backroom, moved around and handled by retail staff, there are more opportunities for “shrink.” The severity of the problem is directly linked to a retailer’s shopkeeping standards. This means making sure that all products arrive undamaged at their correct locations. This applies to moving products from the back room to the store shelf without losing them along the way, damaging them along the way, or misplacing them at the final destination (i.e., hiding them somewhere else on the shelf, *Hidden on Shelf* in Figure 3). It also applies to moving excess products from the store shelf to the back room without losing them along the way, damaging them along the way, or misplacing them at the final destination (i.e., losing them somewhere in the back room, *Lost in Back Room* in Figure 3). In the OOS model in Figure 3, notice that there are variables that indicate the “level of organization” exhibited at the retailer (e.g., *Back Room Organization Standard*, *Shelf Organization Standard*). These variables range from 0 (terrible, everything misplaced or lost) to 1 (great, nothing misplaced or lost) and can have a significant influence on OOS. Intuitively, the more precisely a store can know exactly how much of a product is where in a store, the better chance that store has of meeting current customer demand and ordering and receiving the right quantity of product to meet future customer demand.

Figures 15 and 16 highlight this phenomenon. Again, starting from the baseline case as in Figure 4, demand and ordering are set at 20 units/hour and the *Shelf Slot Capacity* is set at 30 units. However, we have added 10% shrinkage to inventory in the *Back Room*, indicating that at all times 10% of inventory in the *Back Room* is permanently lost, stolen, or damaged (and cannot be returned to the manufacturer). In Figure 15, the amount *In Hand* and on the *Shelf* is limited to 18 units/hour because the incoming order rate of 20 units/hour is reduced by 10% (i.e., 2 units) in the *Back Room*. Demand still remains at 20 units/hour. Because the *Shelf* started with 30 units, the additional units on the *Shelf* allow the retailer to meet demand for the first 6 hours. After that, the retailer is only able to provide products at the rate of 18 units/hour, so lost sales are accumulating at the rate of 2 units/ hour. Figure 16 shows the

Addressing Retail Out-of-Stock Issues Using System Dynamics

total missed sales increasing at a rate of 2 units/hour beginning in Hour 7 of the simulation. As with the previous scenario in Figures 12 and 13, this represents a chronic OOS situation.



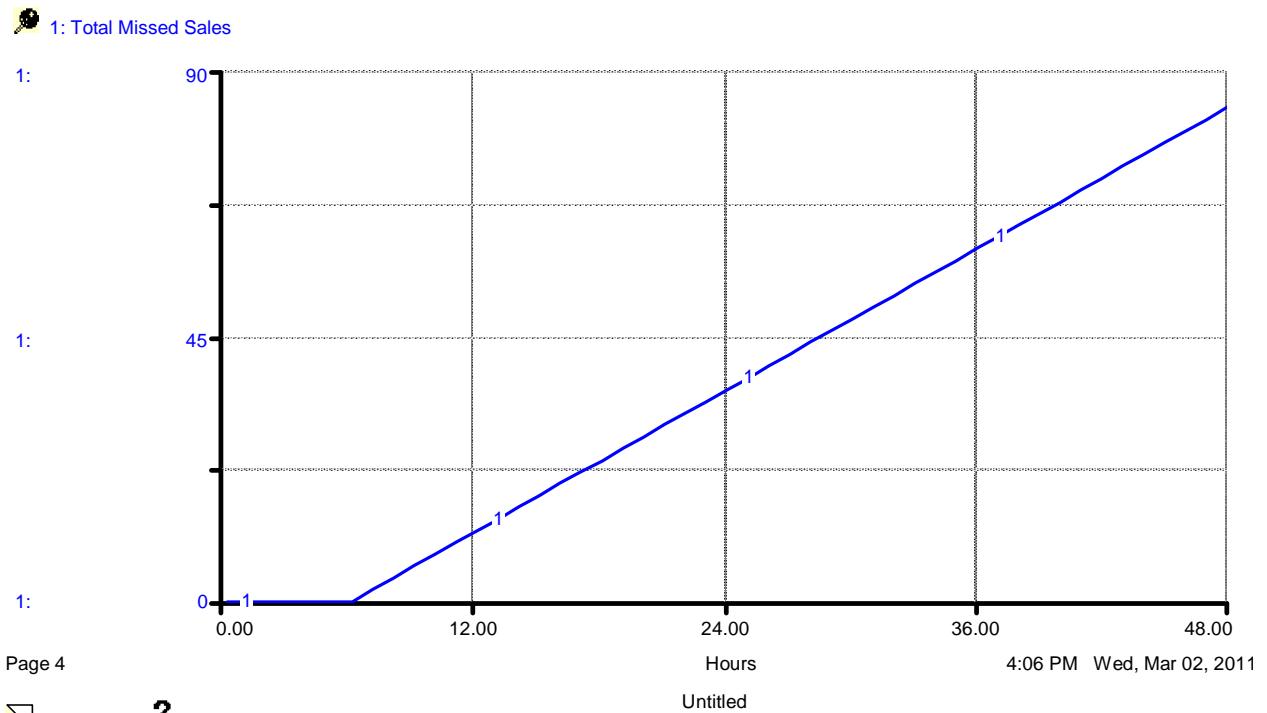
Page 1

4:06 PM Wed, Mar 02, 2011



?

Figure 15: Demand = 20, Shelf Slot Capacity = 20, Back Room Shrink = 10%



Page 4

4:06 PM Wed, Mar 02, 2011



?

Figure 16: Demand = 20, Shelf Slot Capacity = 20, Back Room Shrink = 10%

Shrink also is a major contributor to “phantom inventory”, which refers to the discrepancy between a retailer’s inventory record in its Point-of-Sales (POS) system and physical inventory in its stores. With shrinkage, the retailer POS system may think some products are available because they have been checked into the system during the receiving process from the dock. However, in reality the products are not available for purchase. Phantom inventory can lead to prolonged OOS situation if not corrected. If the retailer POS system says that products are available (but they are not), then it will not reorder the product for replacement. Thus, OOS continues. The greatest peril of persistent OOS is that the store is basically training its customer to shop elsewhere.

Furthermore, a retailer’s automated ordering system is oftentimes based on actual sales. This process can exacerbate the OOS problem. When there is an OOS situation, actual sales will be lower than actual demand. As the ordering system captures lower sales volumes, lower order rates result. When lower order rates occur, fewer products are delivered to the store, which can result in even more OOS situations (if demand is still high). If not corrected in time, it becomes a vicious cycle as frequent OOS leads to actual lower demand.

These insights exemplify a common dynamic in the SD world: A solution to the original problem may only make the problem worse in the long run. In the case of OOS, the desire by manufacturers and retailers to increase customer satisfaction by increasing product variety results in less available shelf space allocated to each product at retail stores. Less shelf space calls for more products stored in the backroom, as well as more frequent movement of products from the backroom to the shelf, both of which are conducive to higher shrink. Because retailers are also pressured to keep costs low, adding labor costs to move inventory frequently is not often done. This fact, along with higher shrink and the resultant phantom inventory, leads to more OOS incidences. Higher OOS leads to lower customer satisfaction. Figure 17 shows this unfortunate reinforcing feedback loop. [NOTE: Product variety is not the only method for manufacturers and retailers to increase customer satisfaction, and OOS events are not the only determinant of customer satisfaction. Yet, within the scope of this modeling effort, this feedback loop holds true.]

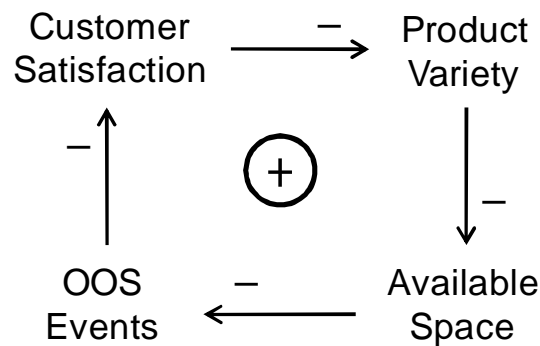


Figure 17: Causal Loop Diagram Showing How Efforts to Increase Customer Satisfaction May Reduce Customer Satisfaction

Translating a SD Model into a Business Tool: RetailBLOX

As with any complex process, simulation is required to help quantify trade-offs and consequences of decisions and actions in the short-term and the long-term. For each specific combination of products

and retailers, a different solution may be required. There is definitely not a “one size fits all” solution. To compound this complexity, multiple products by the same manufacturer are typically sold at a single retailer. And, each product may be sold at multiple retailers. For example, in some situations, it may be worth it for both the manufacturer and the retailer to add labor to move inventory frequently. In other cases, however, this may not be optimal or may not even be possible. In some situations, it may be useful to reallocate shelf space to favor a more popular product and in other cases it may not. Understanding which solutions are optimal for a given combination of products and retailers requires developing a business application focused specifically on that goal.

Therefore, we translated the OOS SD model into a business tool called RetailBLOX, a SimBLOX-based business application. RetailBLOX will be able to “tune” the model to a particular product and/or retailer (or a group of products and/or retailers) and thus identify, analyze, and address specific problems in the delivery, storage, stocking, and sales in their environment. Mostly, the tool will be used to show how recommended changes will make positive impacts on the system or to show how some actions may cause negative impacts and degrade the level of performance for the system. As such, we can use RetailBLOX to highlight and optimize actions and decisions within the scope of responsibility for CROSSMARK, as well as use it to help clients better understand the role that their actions and decisions play in the total system and the resulting bottom line: selling more products.

This type of macro-level model building is not possible in current SD tools. Currently available SD modeling tools (e.g., iThink, Vensim, Powersim) are excellent at developing models on the level of the OOS model shown in Figure 3. In fact, iThink was used specifically. However, these tools are limited when it comes to building large-scale, dynamic SD models because this is not their main purpose. For larger models, these tools become cumbersome, difficult to manage, and prone to error due to large amounts of manual manipulation. While some features, such as arrays in iThink and subscripts in Vensim, are designed to help “replicate” model structures, these features only allow exact duplication of the same structure in a “parallel” process (e.g., Product A and Product B moving through identical processes, but with different cycle times or inventory levels).

As an example, consider the simple Beer Game model. Even though the model structure for a warehouse is generic, arrays or subscripts cannot be used to connect the Wholesaler to the Distributor to the Retailer. Three copies of this generic warehouse structure are created and manually connected together on the modeling interface. Each time this replication occurs to build larger supply chains (e.g., 10 warehouses in a row), this copy-paste-connect process must be used. For complex warehouse models, there may be several variables that need to be connected each time. This opens a lot of opportunities for human mistakes, such as not making an appropriate connection. Not to mention, the model gets bigger and bigger, which introduces new issues related to navigation, visibility, and management. If each individual model for a warehouse is 3 pages, then connecting 10 of them together creates a 30-page model, plus any other additional model structure needed for aggregate bookkeeping, etc. Furthermore, as “dynamic” as we think SD is, this large model is now “static” with regard to structure. If the user wanted to change the order of the warehouses, add more warehouses, or remove some warehouses, a good amount of additional modeling would be required. Structural changes cannot be done “on the fly” at the macro-level.

SimBLOX is a commercially available tool that is designed to allow creation and easy management of macro-level models, including dynamic structural changes at a high-level. It uses a “building block” format in which simulation models represent the building blocks of a larger, more complex system. In the case of the Beer Game, a warehouse model is a building block (represented by an icon) that can now

Addressing Retail Out-of-Stock Issues Using System Dynamics

be dragged-dropped onto the model layout and then easily connected with other building block models (i.e., icons). Model visibility and management are greatly improved while still keeping the underlying model structure desired (i.e., warehouse SD model). In SimBLOX terminology, the building block models are called SimBRIX and are represented by icons (Figure 18). Figure 19 shows an example of a SimBLOX interface for the Beer Game on Steroids that was presented at the 2009 SD conference.

Simulation “agent” model

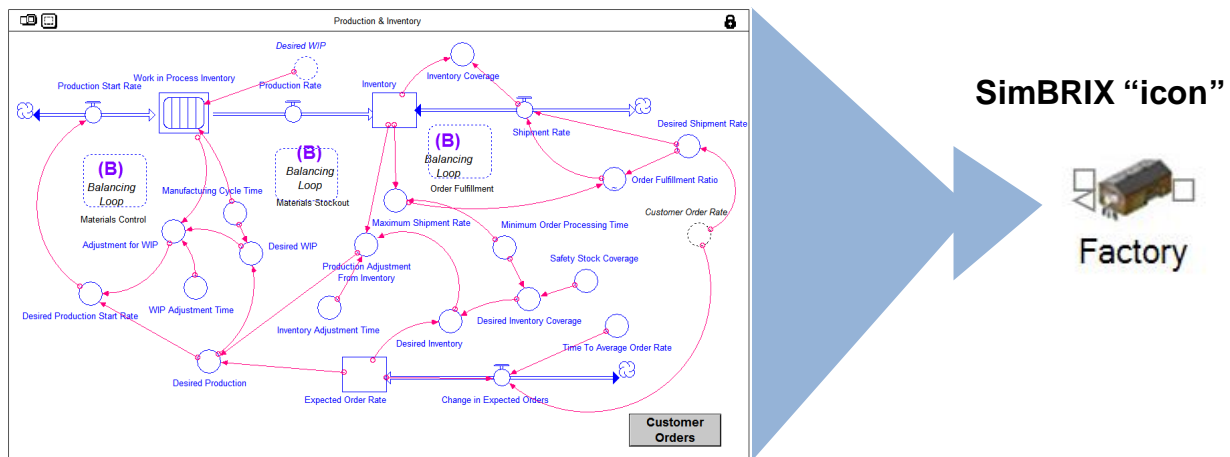


Figure 18: SimBLOX Methodology

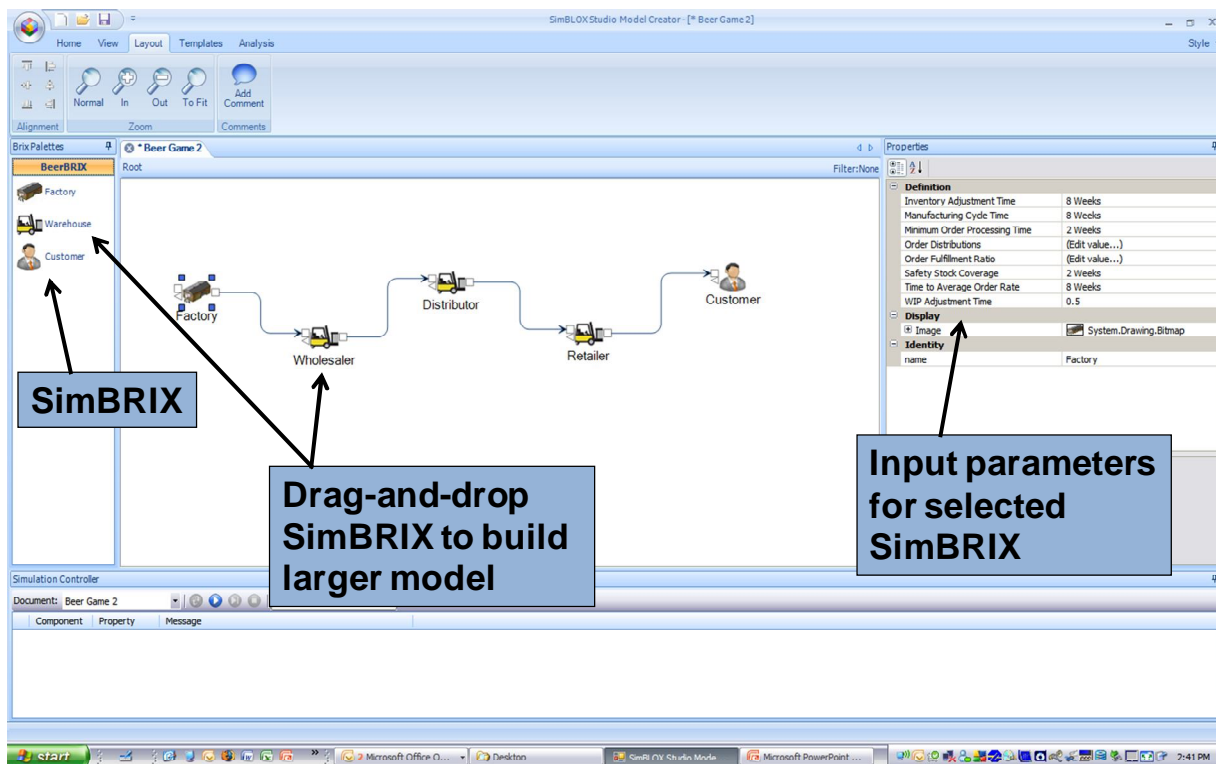


Figure 19: SimBLOX interface for the Beer Game on Steroids example

[Please refer to www.beergameonsteroids.com in which the fundamental building block model is John Sterman's "widget" model from Chapter 18 of *Business Dynamics* (Sterman, 2000), used with permission.]

Figure 20 shows the RetailBLOX interface. The icons on the left side of the screen now represent the various building block models that can be used to build a retail product distribution model and include: Product, Warehouse, Customer, and Retailer. However, in RetailBLOX, there is greater flexibility as well as the ability to increase complexity. For example, the example in Figure 20 shows three (3) retail stores (*Store #110*, *Store #210*, *Convenience Store*) that receive products from two (2) warehouses (*Regional Warehouse*, *3PL Warehouse*). There are three (3) possible products (*Soap*, *Detergent*, *Energy Drink*) and there are aggregate customer demands for these three products (*Demand for Soap*, *Demand for Detergent*, *Demand for Energy Drink*). Note that each retail store can carry a different product mix. In the example, the retail store called *Store #210* is currently highlighted and in the lower left it shows that the store is carrying *Soap* and *Detergent* (only two of the three possible products have their checkboxes checked).

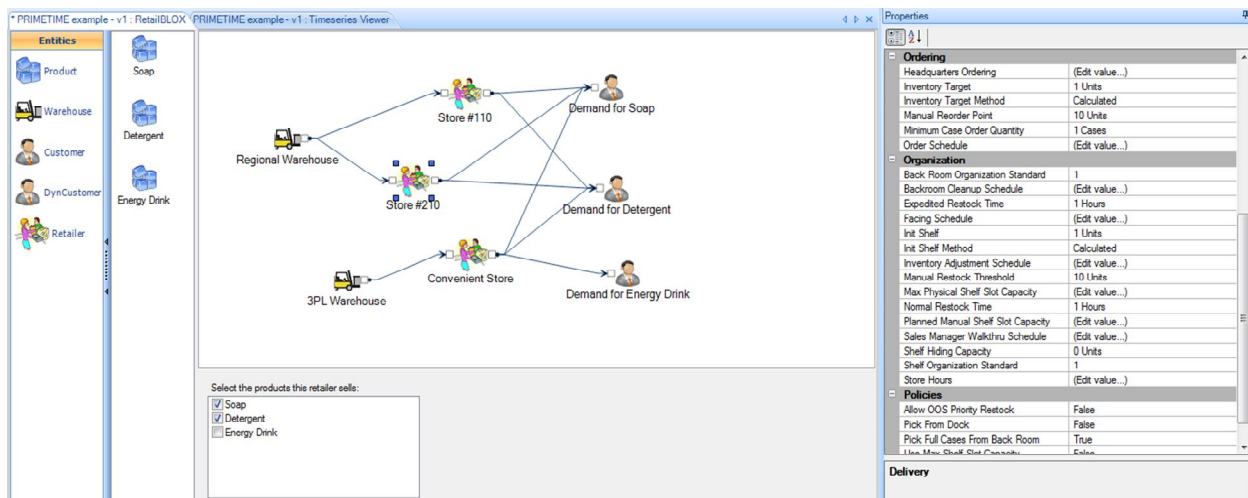


Figure 20: RetailBLOX interface

Behind the scenes, RetailBLOX is creating a series of SD models and aggregating results whenever necessary (e.g., store level) to simulate this situation. For comparison, the OOS model in Figure 3 is a very simple model with only one product flowing through one warehouse and one retailer. For each product, warehouse, and retailer the amount of models is multiplied. So, in the example in Figure 20 with three products, two warehouses, and three retailers, there can be up to 18 ($3 \times 2 \times 3$) equivalent SD models generated internally within RetailBLOX. Imagine copying-and-pasting the model from Figure 3 18 times and manually making all necessary connections to indicated which products are sold in which retailers, etc. RetailBLOX not only hides this complexity from the user, but it also provides an improved user-interface that reduces visual clutter and management of entities in the model.

Results of Test Cases (forthcoming)

Currently, we are putting the final touches on RetailBLOX and identifying example products and/or retailer situations to use as test cases. Thus, we do not have specific simulation results to show at this

point, but the presentation at the SD conference will have results from at least one test example to show the fully functional RetailBLOX and how CROSSMARK is using a SD-based business tool to solve real world out-of-stock issues in the CPG industry.

References:

- Anderson ET., Fitzsimons GJ., and Simester D. 2006. Measuring and mitigating the costs of stockouts, *Management Science*, 52(11): 1751-1763.
- Corsten D. and Gruen TW. 2003. Desperately seeking shelf availability: an examination of the extent, the causes, and the efforts to address retail out-of-stocks, *International Journal of Retail & Distribution Management*, 31(12): 605-617.
- Gruen, Thomas W., and Corsten, Daniel, "A Comprehensive Guide to Retail Out-of-Stock Reduction in the Fast-Moving Consumer Goods Industry," Procter & Gamble research study, 2008.
- Sterman, John D., *Business Dynamics: Systems Thinking and Modeling for a Complex World*, Irwin McGraw-Hill, 2000.
- www.beergameonsteroids.com
- www.simblox.com