Modeling Heuristics

A Major Qualifying Project Report

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements

for the Degree of Bachelor of Science by

Kevin Piala

Date:

Abstract:

The Sterman Beer Game has been shown to be an excellent model of supply-chain behavior. For this project I have modeled several known heuristics in an attempt to describe how humans decide upon their responses. Heuristics were modeled using the system dynamics software Vensim then compared to human responses. It was found that both a cautionary response behavior ('Better-Safe-Than-Sorry') and a complicated anchored-response behavior using all heuristics (the 'Unified' Heuristic) did very well at explaining and representing human behavior. Further work might attempt to use these heuristics to predict human behavior under different conditions.

Table of Contents

Introduction

In supply-chain management the most fundamental aspect is the behavior of the supply-line. Without understanding how the supply-line behaves, decisions about ordering are nearly impossible to make well. Currently, there has been a great deal of mathematical modeling of the behavior of a physical supply-line. This enables people to understand how various delays will influence the behavior of the system. The objective of this MQP is to further examine the human
behavioral side of the supply-chain.

In order to better understand human behavior, it is important to be able to simulate it in some way. While their are many potentially means of simulation, computer simulation was used because of its repeatability and clear results. There are many varieties of potentially applicable modeling. Agent based modeling focuses on discrete events between working characters, but would be inappropriate because the rigid structure of the game defeats the purpose of the technique, and because it is the decision-making structures themselves that will be focused upon. Instead, System Dynamics modeling was used for this MQP, in order to simulate continuous thought in a game with a rigid structure.

Additionally, rather than attempt to create a supply-chain and data from scratch, the Sterman Beer Game was used. The beer game, is a supply-chain simulation game. It exists as a tool which allows players to see the implications of a delay when attempting to correct for discrepancies, has a long history of use, and a significant amount of data from real runs of the game. The beer game was transformed into a system dynamics model and various well known decision-making methods, called heuristics, were used as examples of how the players might be making their decisions. Each of these examples was then tested independently in order to determine if their behavior matched the basic behavior of real humans playing the Beer Game. The end-goal of this MQP is to discover a heuristic which better explains human behavior than the current accepted method proposed by Sterman.

The Beer Game

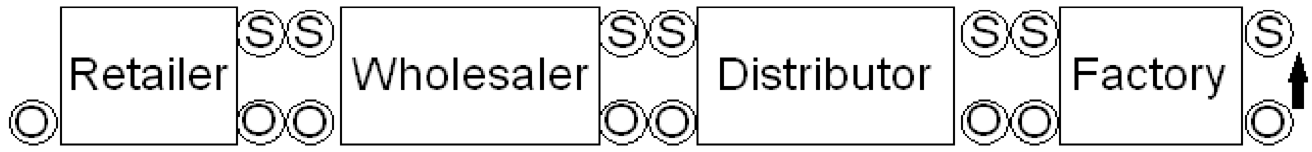


Illustration 1: The basic beer game

The Beer Game is quite simply a game in which players order 'units' of product from one-another. Each player is given a sector to control, either the Retailer, Wholesaler, Distributor, or Factory. It takes two turns for both a shipment of supplies and a turn's orders to reach the other sectors above and below the player in the supply line. The game has been played with a number of interfaces, but the most common way is by playing on a mat similar to the illustration 1, shown above. Players are only able to see their own inventory and the supplies and orders they receive on a given turn. The game is fairly simple when drawn out. Because the players see orders and shipments only when they reach their sector there is a delay in the possession and knowledge of the supplies. This increases the system's instability. Even in a perfect knowledge system, players tend to act in a non-ideal fashion, making it highly unlikely that they are covering the game in an entirely logical manner(Sterman, 1987)

Beer game runs are unusually consistent. When looking at data from Students, professors, and businessmen it can be seen that the same basic pattern is extremely common. This 'normal' pattern can be seen in almost every run to a certain extent. This 'normal run' shows an exponential increase in the variability of ordering. There is an effect classically described the 'bullwhip effect' when players order goods across multiple sectors. Their orders tend to multiply with ever successive sector, without a change to the final desired goods.. The result is incredibly instability and generally poor behavior. (Drezner, 2000)

The Beer Game: Normal Play

There is a wide variety of response in normal play of the beer game. Any given game run can result in anything from near-ideal behavior to catastrophic failure. That is not to say that there isn't a

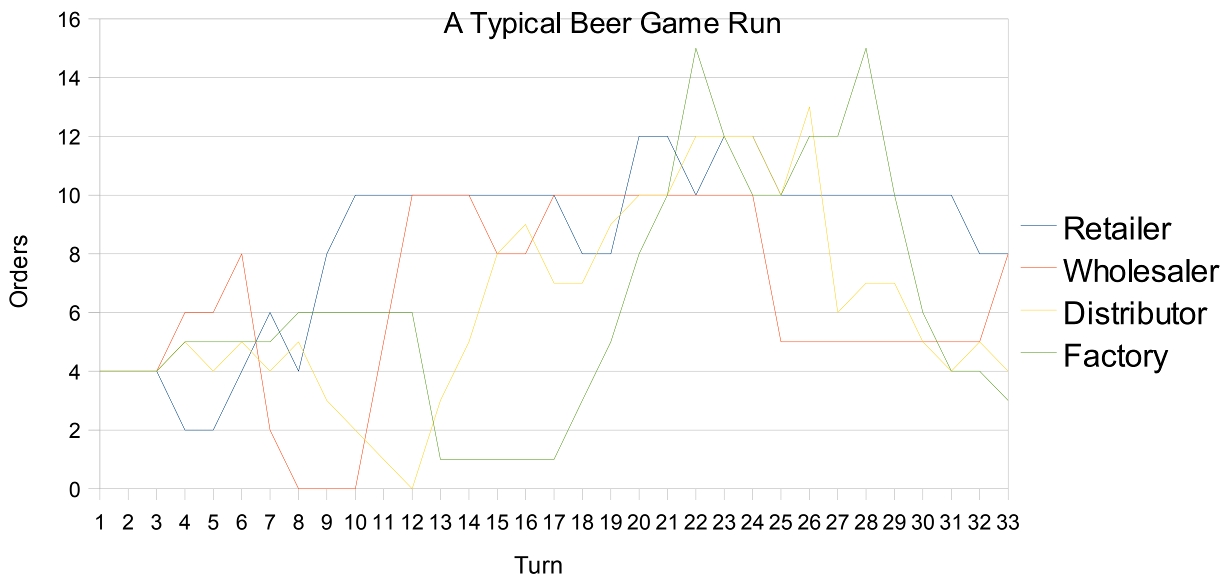


Illustration 2: Ordering during an average beer game run

typical set of responses. The above graph, Illustration 2, comes from the original study done by Sterman (1986) and is an excellent example of failure. You will notice the bull whip effect in the ordering patterns, and no responses seem very poor until you look at the complementary inventory graph. The distributor is so reluctant to order additional stock that they become incredibly ineffective. The retailer suffered similar problems. Admittedly in this case it was very close to ideal ordering patterns in the beginning, but small deviations caused massive problems. The bull-whip effect caused by this can be seen very clearly in the later oscillations of ordering and the long term implications on inventory. Overall when one compares this to other player responses, the basic pattern can still be seen. The largest difference is generally a change in phase shift for a given sector and the resulting changes in inventory.

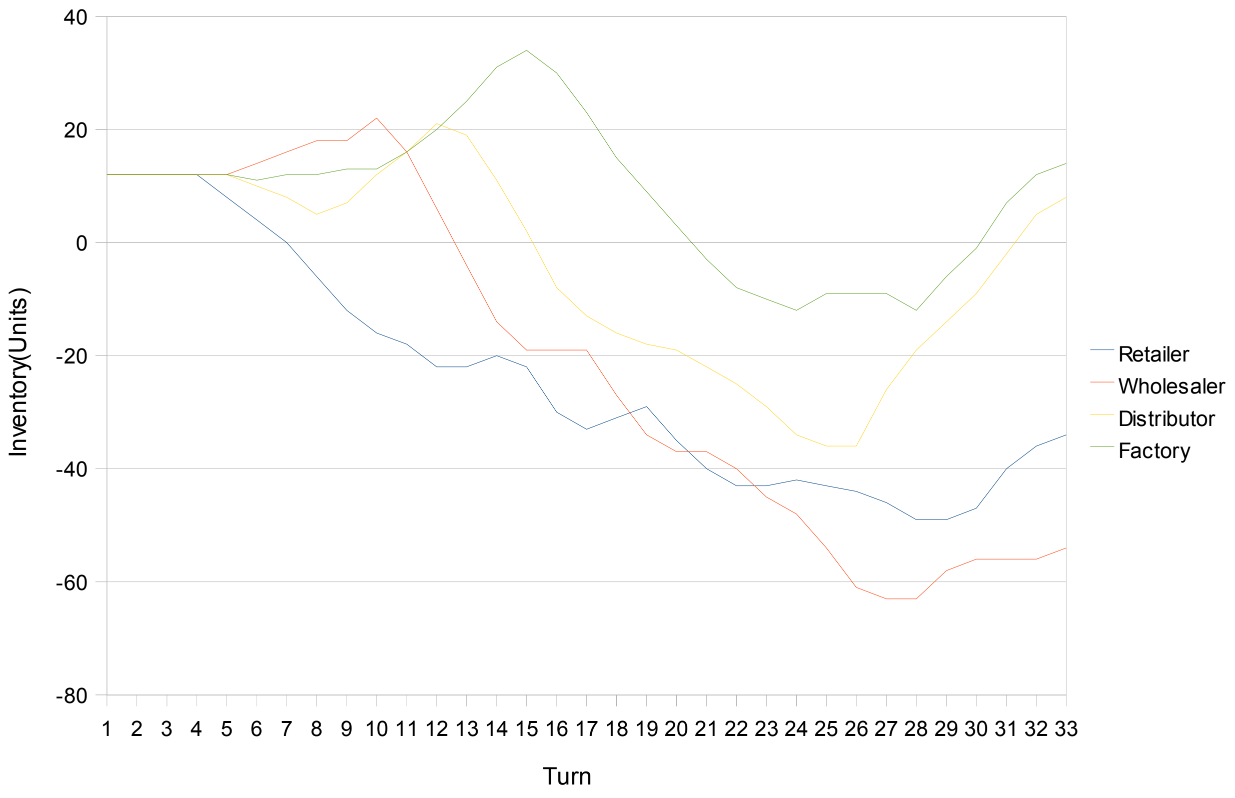


Illustration 3: Inventory levels during an average beer game run

Normal play of the beer game by humans has a few notable aspects. There is usually a rolling wave of increasing orders that cause significant backlogs, and there is usually one more oscillation repeating the cycle of too much or too little inventory. The reasons are debatable, but usually are related to players being unable to appreciate the supply line or being unable to react to spikes in orders and overreacting because of this. Players usually do not play the game well, and even those who know exactly how the game normally plays out still do not often play well. Out of the initial eleven runs, ten of them had two or more sectors with a backlog greater than 40. Of these just over half then proceeded to shoot up to an inventory greater than 40.. The single run of the game that did not experience these effects had a factory sector that had an inventory exceeding 80 during the start of the game. Below, Illustration 4 is a stereotypical beer game run.

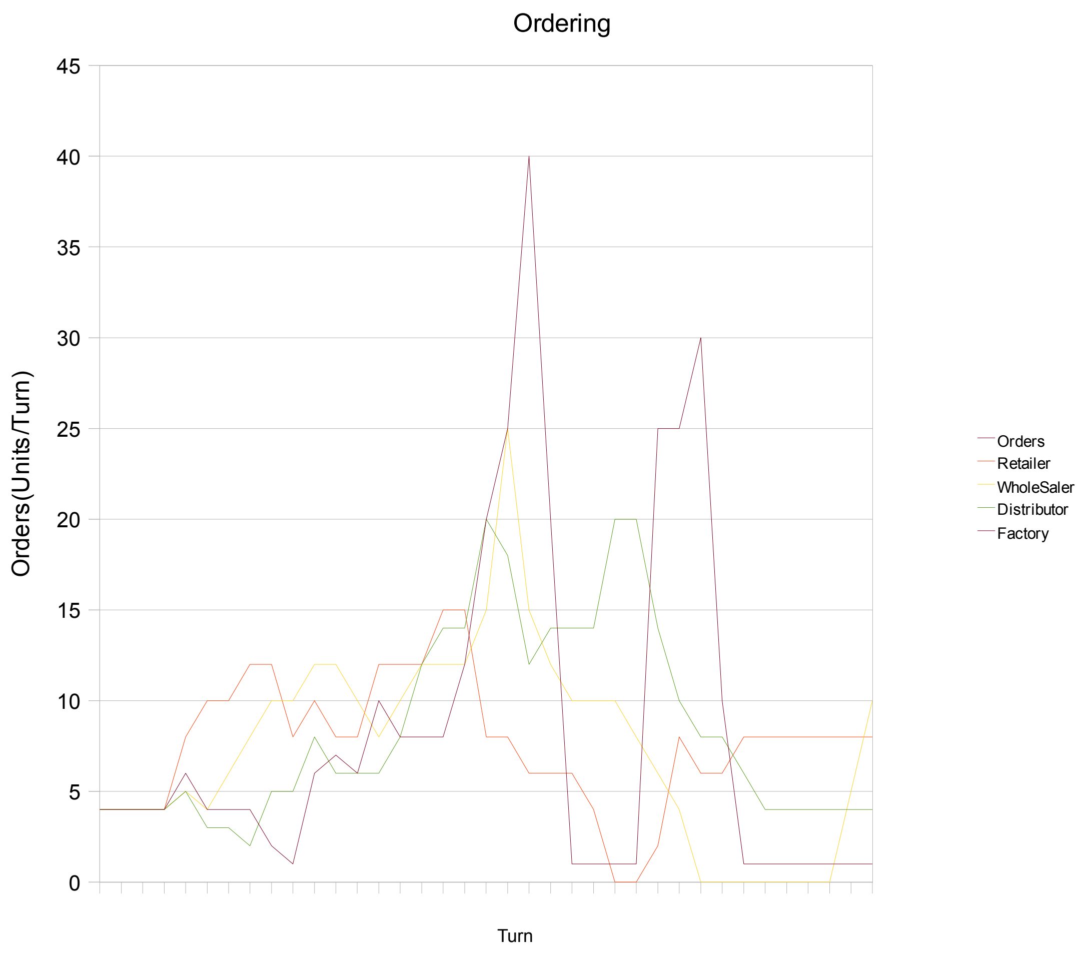


Illustration 4: What failure usually looks like

There is a great deal of change of variability of order. One can easily see the heights of the ordering patters following a pattern, a phase shift to the right for those groups later in the process and an increase in amplitude. This is a simple response to a rapidly increasing backorder. The factory generally has the lowest increase in amplitude for a number of reasons. It is both in control of its own shipments directly and subject to a smaller delay than other sections. When the player knows this, they generally have a cleaner response to their requests and are less likely to over-order.

This tendency can be seen in graphs because the factory generally does a very good job at keeping its inventory close to the desired amount, a better job than the wholesaler and distributor. The retailer holds a similar advantage because it is dealing with a more constant demand. It is reasonable to assume that it would have a more variable inventory if the customer demand was less predictable.

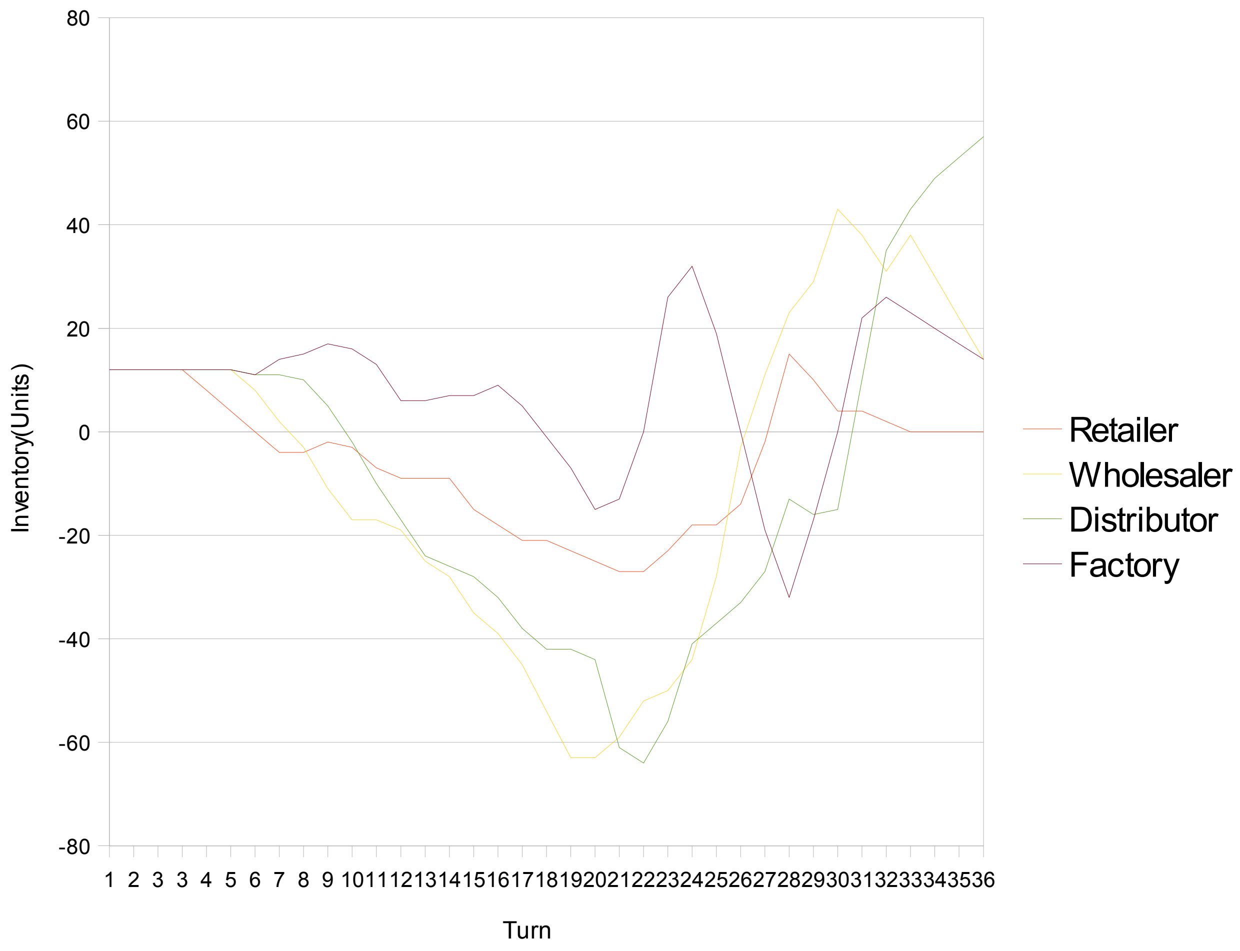


Illustration 5: Inventory levels during failure

It is important to note that it is functionally impossible for the retailer to avoid a 'negative' inventory when the game begins. Individuals playing the game are instructed to order 4 of everything for the first few turns and have no way of know if or when there will be an increase. Because of the delays involved with the game, it will take at least four turns to receive a new order. This means that unless the player already knows the increase is coming, he will be unable to his shipments before his inventory of 12 depletes over 3 turns. This means that there will always be an observation of behavior during a backlog. While in an ideal response case the players can minimize this to -4, it is unlikely that this will occur.

The goal of my work is to discover a different way to describe the decision-making process of players, a way which might yield better insight into why players act poorly and how to correct their personal problems. In 1987, Sterman first publicized his results of a study upon the game. He created a series of equations to describe and simulate the behavior of players. It is known that Sterman's system when calibrated correctly can accurately simulate the results of most players, therefore instead of competing with this philosophy the other heuristics are present merely as differing indicators of what bias is likely occurring. Originally, the goal of the tests was to determine whether the heuristics presented could explain the data better than the equations presented by Sterman, however this was determined to be unrealistic. Instead, it was found that Sterman's equations were overly robust and would maintain their net variability regardless of initial conditions. The goal soon became to find a heuristic that could explain normal behavior and find an explanation for what might occur during more unusual circumstances. The theory was that if the model can account for both the best part's of Sterman's model and surmount the weakness' of the model it might then provide better insight.

These effects have already been calculated and discovered statistically and logically by a number of researchers like Sancar. Unlike the traditional view on these problems, one can try to figure out why people make these decisions. The classic viewpoint is simply that players do not account for the supply-line, that players do not account for delays between the cause and effect of their actions (Sterman, 1987). This is without a doubt true, however currently established theories do not properly describe why even smart players fail to account for the supply (Kleinmuntz, 1985). There is no focus on the underlying reasoning of these actions. By focusing on human reasoning, we can better understand the implications of these problems on the large-scale, and where these problems will most likely occur

Prior Research

A great deal of research has already been done on the Beer Game, supply chain management, and a number of related topics. The findings cover everything from the risk taking tendencies of certain game players to the fundamental cognition of children when dealing with adverse situations. The findings can be largely divided into a few different categories. There is that research that attempts to mathematically explain already seen behavior. This was covered in a number of different ways including Sterman's equations for the simulation of the beer game and Sancar's formulas for calculating the bull-whip effect for a given supply chain. There are the studies on human responses to complected simulation, largely done by Kleinmuntz and Elg. Finally, there is a large amount of background research simply into the basic ways humans process data. This background research into decision-making has been the subject of a huge range of experiments by psychologists. All of this information is important for creating a thorough image of what likely occurs when people play the beer game, but none of it directly deals with this question in any detail. The subject has been applied to statistical analysis of performed games and theorized aspects of Supply-Chain management, but never anything as specific as an analysis of the Beer Game itself using these concepts. This means that one must look at research on human decision making and supply chain management separately oftentimes in order to gain a clear picture of how the two areas relate.

First, it is most important to understand the background research on decision-making. In this research, it has been found that the processes involved are generally quite simple. It has been commonly shown and proven that the human mind cannot easily hold more than seven plus or minus 'chunks' of information in short-term memory(Miller, 1956). Traditionally this designation is applied mostly to things that cannot be 'summarized' in our minds. We might remember the seven digits of a phone number or a few fully encoded phone numbers. This restriction is especially notable when the person is attempting to remember pieces of information that are not directly related and then attempting to use them for processing(Cowan, 2000). In these cases it seems the limit is often closer to four areas of information that one can see then use in working memory, without attempting to commit the information to long term memory.

This is especially true in the case of complex situations, where individuals are generally seen to rely heavily upon heuristics rather than processing the entire situation(Brehmer, 2005). It is a common failing of behavioral models to ignore this disability.

Depending upon the situation the heuristics that humans use have been found to be anything from a simple reaction to positive or negative to an anchor-adjust heuristic based off of a known solution (Kahneman 1974). What varies is the sort of heuristic that is applied to a given situation.

The greatest fault and greatest success of the original Sterman model is that it looks entirely at the functional side of the decision-making process(Sterman, 1989). Rather than attempt to understand the human mind directly Sterman instead modeled the components that it functionally represented. It was his foremost discovery that players seemed to under-count the supply-line, either seeming to ignore it or failing to account for it properly. He did not, however, determine why this occurred, simply that it was the simplest explanation for the poor behavior (Sterman, 1987). Sterman's Model was based largely around the functional aspects of Supply-Chain management decisions.

The way Sterman separated the decision-making process was to break it down into several components. The first component was the incoming orders. Sterman theorized that players would always base their orders off of the incoming orders. In his eyes players would not under or over-order based off this information alone. They would instead start their calculation from the new incoming orders. They might expect more or less orders depending upon how their expectations were formed, but the foundation is fundamentally the same. The second component was the inventory adjustment rate, represented by the constant 'alpha'. This variable represented the rate in which players would correct for effective inventory discrepancies. A player with an 'alpha' of 1 would order an additional amount equal to his inventory shortage each turn. This affects the rate at which they adjust for supply-line discrepancies as well. Finally, there was the percentage of the supply-line they accounted for, beta. According to Sterman's calculation, a player consciously or subconsciously determines his desired supply line and inventory, his current supply-line and inventory, and adjusts the incoming orders up or down after determining the difference(Sterman, 1987).

In this manner, he has managed to create a comprehensive way of simulating human behavior during the Beer Game, but in order to gain any understanding on humans one must work backwards from it . The model gives no inherent understanding as to why the players are making these decisions. By his original model, an individual would need to be processing information in an extremely unlikely way, keeping track of every variable specifically and coming to a full decision of what to do each round. There exists no bias towards previous decisions, and players would come to the same decisions whether looking at the situation for the first time or as it arose during the middle of the game Sterman's model also fails to explain all situations adequately. When a group plays the Beer Game with perfect knowledge, they still run into the same fundamental problems, even if they have been given excellent data and have had the optimal strategy explained to them (Sterman, 1989). While Sterman's model can simulate players behaving poorly when they are aware of the inner workings of the game, it does not explain why they would continue to fail.

Independent research has given us some excellent insight into what people might be doing when making their decisions in these situations. It has been shown quite thoroughly that people generally take a complex situation and make it extremely simplified(Brehmer, 2005). In complicated situations especially, it has that shown that they avoid fully processing the situation(Kleinmuntz, 1985). In Elg's experiments(Brehmer, 2005), they tested to see if subjects were able to deal with a simple simulation in which fire fighting units (FFUs) were properly allocated to a fire, taking into account the time constant of the travel time of the FFUs from the start-location. The players seemed to do very well when they were able to position themselves such that they did not need to distinguish between the types of fires. All cases could be handled in approximately the same way by dividing resources across the area. When they could not minimize the presence of the time constant in their decisions, they performed much more poorly. Players could no longer cope with the situation well and began to send too many or too few FFUs to a location regularly. The implications of the experiments are quite interesting. It was found that subjects would tend to respond poorly to time constants while assuming the worst-case for those sections that they could not easily process. Rather than distinguish accurately or inaccurately whether the fire required multiple FFUs, they instead determined if they were certain it only needed one and sent two if they could not be sure. Basically, in this case it shows that players will respond to time constants, but poorly. When interpreting the time constants on the larger scale they will tend to 'round up', trying to avoid the 'failure' condition, even if there is no specific cost for failure(Brehmer, 2005). In the beer game this would equate to players having a poor memory of the supply chain, but assuming it was insufficient in number unless it was clearly above. Structurally, this would produce very similar behavior, but it gives some insight that the player does not process the situation perfectly in a complex situation, but rather uses a simple system for judging further decisions.

Besides Sterman's original work, the most remarkable attempt to simulate the decision-making process of the Beer Game was that done by Don Kleinmuntz. In this case, though, the testing was less robust than that of Sterman's model, only covering a more general and basic case rather than a fully parametrized estimation of true data. This model, produced by Kleinmuntz was based upon the concept that players are generally fairly simple in their approaches(Kleinmuntz, 1993) . His concepts are based upon the concept that in most dynamic situations it is highly unlikely for a subject to apply a complex formula to the problem. Generally, dynamic situations are advantageous because of their feedback. Someone can easily notice their mistakes and rectifying them, greatly increasing accuracy as the situation continues. This is the case for most dynamic tasks in reality and it relies heavily upon clear and obvious feedback. This is not the case in many complicated situations, such as the beer game and the above fire-fighting simulation where large time constants hinder feedback.

Kleinmuntz's explanation for the tendency for humans to behave poorly was by the concept of the rational allocation of cognitive effort. That is to say he hypothesized that someone playing the game would rationally choose those elements present before them to be most important, and that the existence of the supply line would be less important and received less processing power. In this way it would appear to have a reduced overall impact that normally is present. This theory is interesting because it takes the theories about incomplete supply-line calculation from the traditional model and simplifies it into a short-term process, requiring half as much thought as the alternative, such that it could be maintained on a largely subconscious or reflexive level.

The reasoning explains why players under-count the supply-line (but not other aspects) in Sterman's example. Not only is the supply line a delayed process that takes place over many turns, but there is not specific indicator of it during the game. Players would need to greatly increase their effort to take it into account, and then would only slightly increase their performance.

He goes on to explain how another possibility is the incomplete usage of decision making structures. That is to say – the players do not even use a full and proper heuristic for the situation simply because they can't process the situation but because they for whatever reason do not have the will, desire, or thought to make a full decision structure to cover the situation. Instead the players use something far more amorphous, choosing to go with a collection of partially crafted ideas that together form a basic and hastily crafted decision.(Kleinmuntz, 1993) This result is possible, but would involve a substantially different structure than the Sterman model. The result would be an adaptive structure made from other very weakly used and scattered heuristics. A decision like this would likely not represent one from players attempting to rationally win the game, but instead those who wish to do well yet merely commit a very basic amount of thought to it, relying largely on 'instinct'. The final concept proposed was simply to explain the faults by an improper understanding of the situation, though this is highly unrealistic because of how thoroughly the game is explained to players and because of its concrete nature.

In order to explain this, then, one must look towards the set of simple decisions an individual might make during the Beer Game. It has been shown that an individual, rather than dealing with time constants in a simulation, will simply order more than they would realistically need (Elg, 1997). This response of 'better safe than sorry' might work well in many situation, but in a situation like the beer game it will simply cause an explosion of demand leaving individuals on the top with far too high an inventory after an artificially expanded expectation of future orders. In the end moves like this simply work to vastly increase the bull-whip effect, even if players take into account the situation 'properly'. The extent in which this would increase the bull-whip effect is so extreme that the process clearly cannot be solely responsible for their behavior. The player themselves must therefore be aware of some restrictions to play, at least on a subconscious level. Reasons for this vary, but on the simplest level it is likely that an individual has developed a mental model of the situation that emulates the normally analyzed behavior(Kleinmuntz, 1985).Over time their mental model will change and their judgments will evolve gradually toward an 'ideal' situation, but still using simple decision making components. Their decisions, then, are made off changes of some base heuristic that is altered during game-play. What that base heuristic is, though, remains to be seen.

The current state of supply chain management indicates that with any supply chain the variation will follow the general trend of being additive when the chain is centralized, or multiplicative in the case of a decentralized system. This means that for the purposes of the beer game that variation should be multiplicative based upon the number of steps, advancing as it approaches the factory sector. (Sancar, 2003)

Centralized Information
Additive Model
$\frac{Var(q_k)}{Var(D)} \geq 1 + \left(\frac{2\left(\sum_{i=1}^{k} L_i\right)}{p} + \frac{2\left(\sum_{i=1}^{k} L_i\right)^2}{p^2} \right) (1 * \rho^p) \quad \forall k$
Decentralized Information
Multiplicative Model
$\frac{Var(q_k)}{Var(D)} \geq \prod_{i=1}^{k} \left(\frac{2L_i}{p} + \frac{2L_i^2}{p^2} \right) \quad \forall k$
where k is the number stages of the supply chain.

Illustration 6: Sancar's quantification of bullwhip effect

One of the biggest steps to overcome is discovering the ramifications of the use of different heuristics and whether they differentiate much from Sterman's findings. In order for an individual to be accomplished at any given setting, it has been seen that they must adapt to a setting, changing their known cause-effect structure to fit the environment better (Elg, 1997). This is not the case in the beer game, where the casual structure of the game is already relatively clear. Instead, in this 'pure' example, the individuals are free to use more readily available heuristics as a means of solution to the problem.

Most the prior research can be summarized quite simply. People only perform very simple tasks when encountering a complicated system, such as the beer game. These tasks are dependent largely upon a pre-established bias for the situation,Once the situation has been established they then continue to use this very simple heuristic, but adjust their thought process in order to better deal with the apparent complexities of the situation. Overall, in a decentralized system this leads to a multiplicative bull-whip effect with a tendency to over-purchase goods. The one remaining factor is how people deal with very poor situations.

A brief introduction to System Dynamics modeling

"System Dynamics is fundamentally interdisciplinary.... System Dynamics is a powerful method to gain useful insight into situations of dynamic complexity and policy resistance". Most importantly, it deals with "the counter-intuitive behavior of social systems"(Sterman, 2000).

Fundamentally, System Dynamics is a means of simulation that focuses upon feedback loop interaction in complicated systems. There are two facets of System Dynamics modeling that one must be aware of. The first is the high reliance upon feedback-loops.

A feedback loop is simply a means of representing a complicated system. When an action occurs in a dynamic system, it alters the environment, which in turn alters the condition that brought fourth the action. Take for example, a student attempting to heat an apartment in the winter. When the temperature is below the desired temperature, he will increase his space heater's power. If it is above the desired temperature, he will open a window. This in turn increases or decreases the room's temperature over time, bringing it towards the goal temperature. This, in turn, causes the student to lessen his initial reaction, or change the direction entirely. In the end, the temperature will gradually adjust towards the desired temperature, because of the quick responding negative feedback loop. These feedback loops cause most of the behavior in any dynamic system

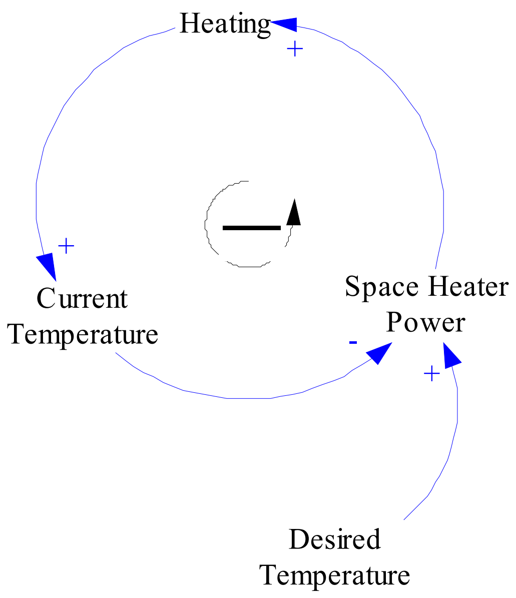


Illustration 7: A basic feedback loop

At its core, System Dynamics is just the calculation of differential equations through mathematical simulation. Simply taking into account the existence of the feedback loops accounts for neither their strength or long term effects. In order to simplify picturing these problems, they are often given a visual representation. The static elements are generally called 'stocks'. These are the numbers that do not change immediately, but are rather the results of flows in or out. For example, a body of water or money could be seen as a 'stock'. This body is changed by the flow in or out of it. A flow is simply the calculated amount leaving or entering a stock. One can imagine this like the process of earning money or water leaking out of a container. The flow is in respect to time, therefore if no time passes they will always be nothing. The units of flow are simply those of the stock being altered, per unit time.

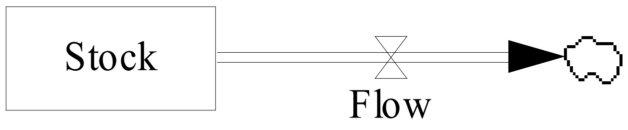


Illustration 8: A basic stock and flow

Another key concept behind System Dynamics simulation is the way in which the flows are determined. A basic system of independent stocks and flows is simply an easily calculable differential equation, but once the system has real-time feedback it no longer can be solved and instead must be simulated. These structures form complicated feedback loops that bring about unusual and often unforeseeable consequences to seemingly simple systems. Because of this, it is important to not just look at the feedback loops themselves, but simulate the full system. That said, simply being aware of the existence of these feedback loops can be tremendously useful when attempting to understand the behavior of complex systems.

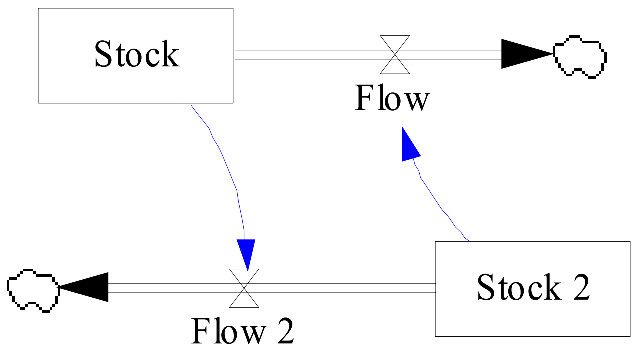


Illustration 9: A more complicated system

Any time-based system that can be expressed mathematically can be expressed with differential equations and therefore simulated using System Dynamics. In order to better simplify systems, each calculation is generally separated so that it can be better modeled. For example, take the most basic possible decision making structure for the Beer Game, which will be discussed later. Let:

Inventory=INTG(Received-Shipped)

Inventory Discrepancy=Desired Inventory-Inventory

Orders=Inventory Discrepancy*Alpha

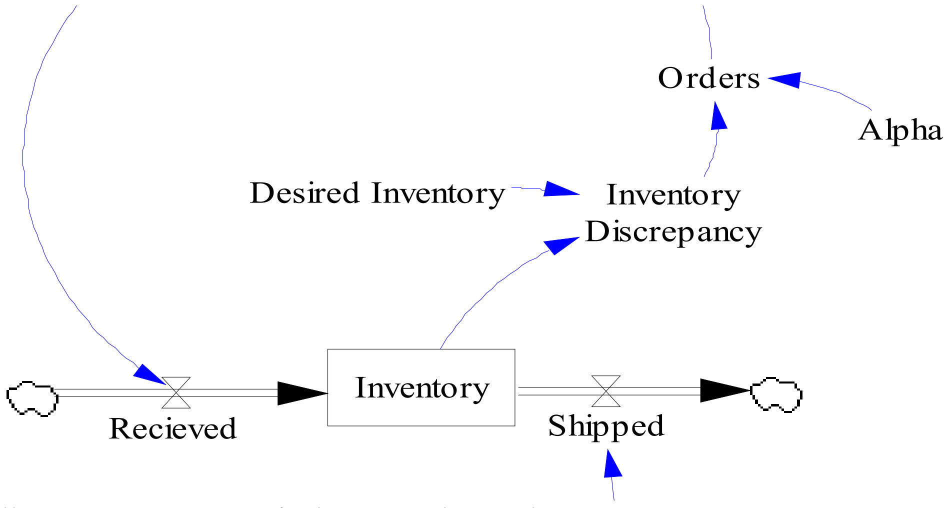


Illustration 10: Part of a larger multi-stock system

The example is just a part of a larger model but serves as a demonstration of what occurs during normal modeling. The implications of these interactions become less clear as larger models are formed. Oftentimes very complicated feedback loops are formed, dictating the behavior of systems. These feedback loops are always present but can often be specifically defined once the model has been created.

The Model

Methodology

In order to determine the relative validity of various heuristics, each given heuristic was modeled using the System Dynamics software Vensim. The origin of any given heuristic varied heavily. The heuristics proposed by Sterman and Kleinmuntz, for example, were created directly from the equations they proposed. Other heuristics, such as the 'Better Safe Than Sorry' heuristic and Anchor-Adjust heuristic were generated based on behaviors described in other works, especially the compilation of known heuristics made by Kahneman and Tversky.

The first step for creating the model was the generation of an overall model of the Beer Game. Initially, during the first phase of the project, a basic model was made to look at overall behavior. Each sector is simply composed of an inventory with an in-flow and out-flow. Just like the actual Beer Game, sectors receive orders and send goods in response. It takes two 'turns' before goods or orders are received from the next sector in the chain. For example, the function for what the Retailer received was:

```
DELAY FIXED( Shipments from Wholesaler, Shipment Delay, Shipments from Wholesaler)
```

After a delay of (Shipment Delay) the retailer would receive (Shipments from Wholesaler) each turn. The initial value for this was (Shipments from Wholesaler).

The first model referenced a 'mental view' sector in order to determine what their orders that turn would be.

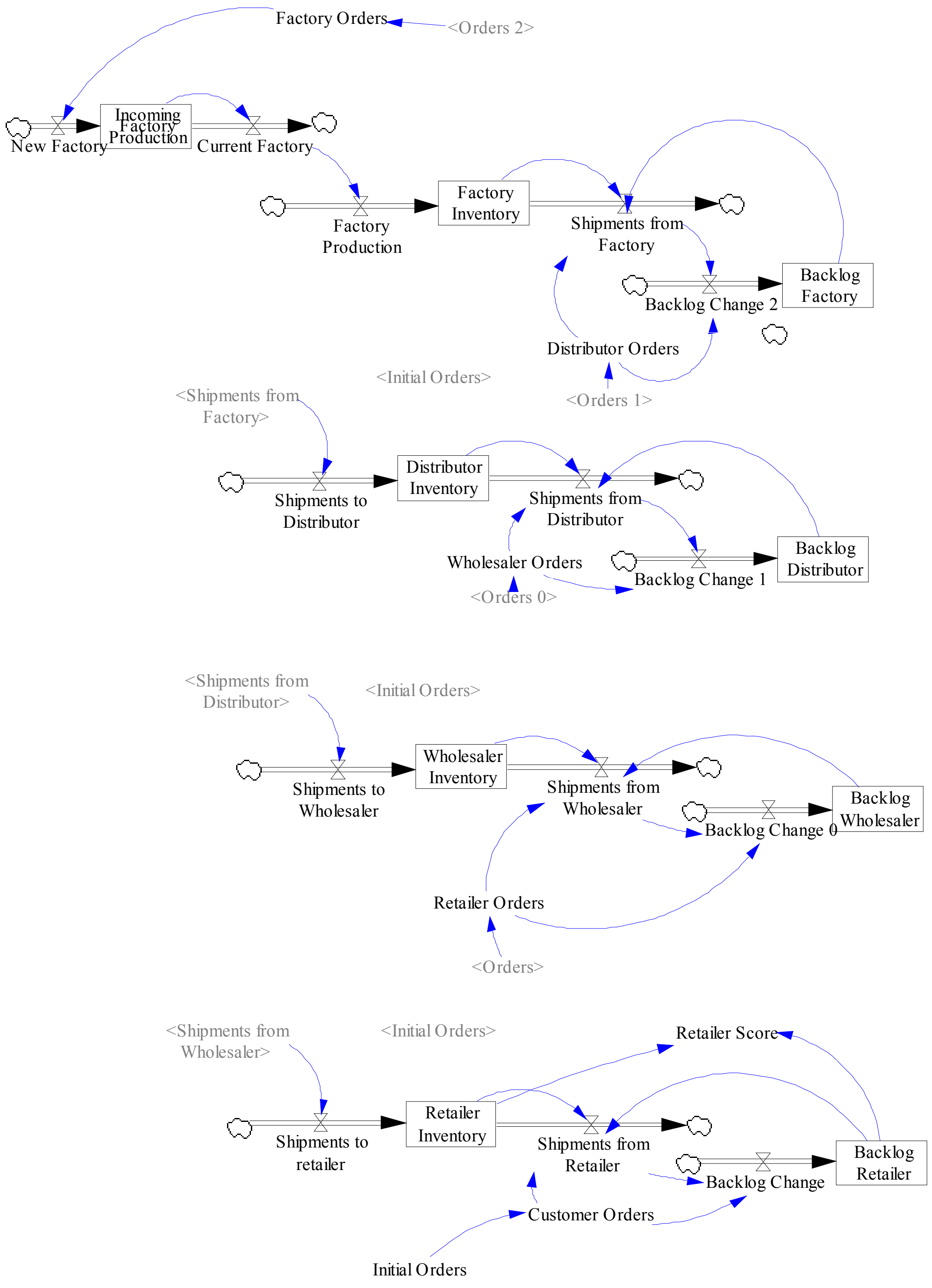


Illustration 11: Initial Basic Beer-Game model

The initial setup was capable of modeling all heuristics that might be designed, but because it was not sufficiently modular it was functionally impossible to look at the behavior of a given sector when isolated. This is important because the isolated sector helps to give a better view as to *why* a heuristic is operating poorly and it because it helps determine if the heuristic is fundamentally flawed in some way.

The second complete version of the model instead began with each separate decision-making structure integrated into the full sector.

For example, the above Vensim model, Illustration 11, is simply an isolated sector of the beer game. It

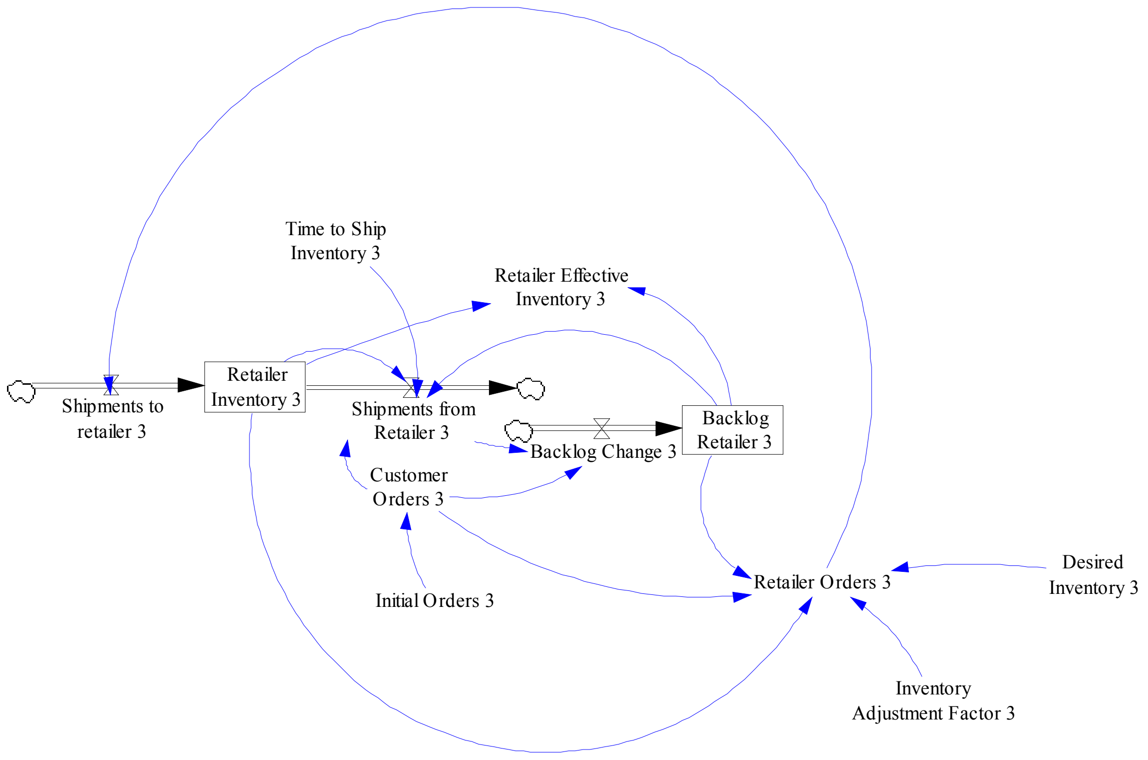


Illustration 12: An isolated retail structure

takes four turns before it receives shipments from its own orders. The delays are therefore always consistent. The result is an isolated behavior that can be compared to other behaviors more completely.

'Validity' is not generally a term used in System Dynamics modeling. Systems, especially social systems, are too complicated to be able to prove that the result is correct(Sterman, 2000). Instead, models are measured for how 'confident' one might be in their results. All of these heuristics successfully pass the first steps in confidence. They are based on known behaviors that humans perform, and they have internally consistent units. Units are an underrated factor in modeling, as they are a required step to show that this process being modeled could actually exist. The final step for determining confidence is their ability to match real data. This can be broken into two sections for these models. First is their ability to output human responses. For this purpose they are simply compared to the human runs Sterman used when first creating his model. The second check is comparing their net variability between sectors. This is compared to see if it is in the same range as normal human responses. The goal of this project is not to find the specific parameters required to output a given human's behavior. It is instead to discover which set of known human heuristics is most likely to be involved in the Beer Game.

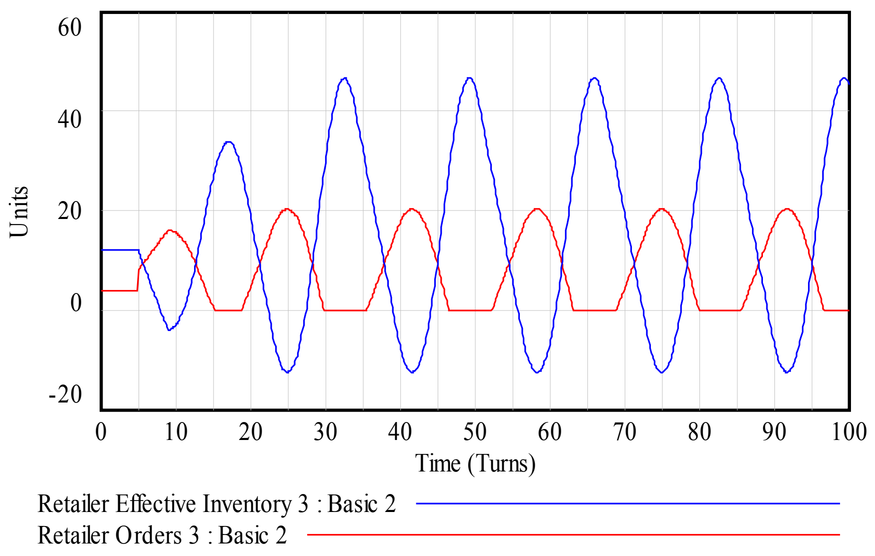
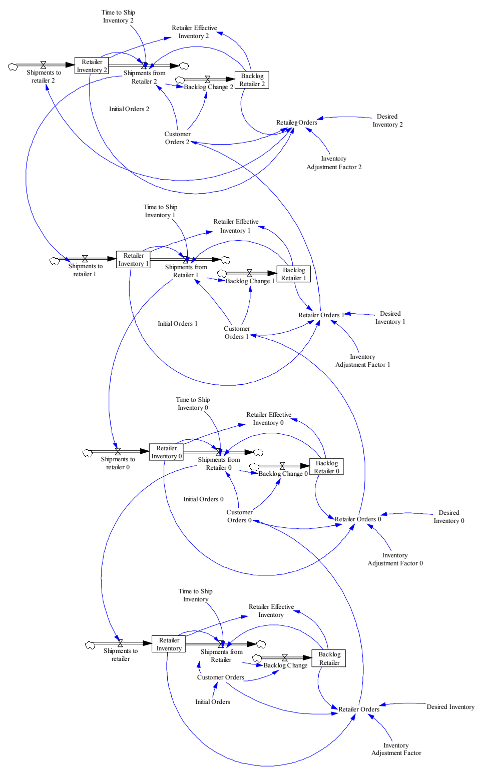


Illustration 13: Isolated Retailer Behavior



Drawing 1: Four Linked Retailers

Illustration 13 shows the basic behavior of an isolated sector in a supply-chain. This example supply-chain is simply experiences a delay of four turns before ordered goods arrive.

In this example there is no real decision making process done, or rather the processes performed is extremely simple. This imaginary sector simply orders as much as is needed at the time, ignoring the supply line and all other problems. This represents what a computer might do in order to automatically bring up inventory levels. There is no thought put into the process and it can be seen to behave quite poorly. For comparisons sake, one can notice that ordering peaks approximately every sixteen turns in this case.

Drawing 1is an example of the structure that occurs when one attaches four of these isolated structures into a full beer-game model. In isolation each of these structures would have the above behavior (except for the factory which only experiences an order-to-shipment delay of two turns). When combined, though, the problems that they experience become multiplicative.

In the below graphs, it can be seen that a simple ordering method in the basic Beer-Game will lead to incredibly poor behavior. The periodicity and magnitude of this behavior is largely based on the time between ordering goods and receiving the desired products. This means that in the full system behavior becomes incredibly sub-optimal. A high inventory for a human player might be 100 units, while this structure can reach over 10 times that number.

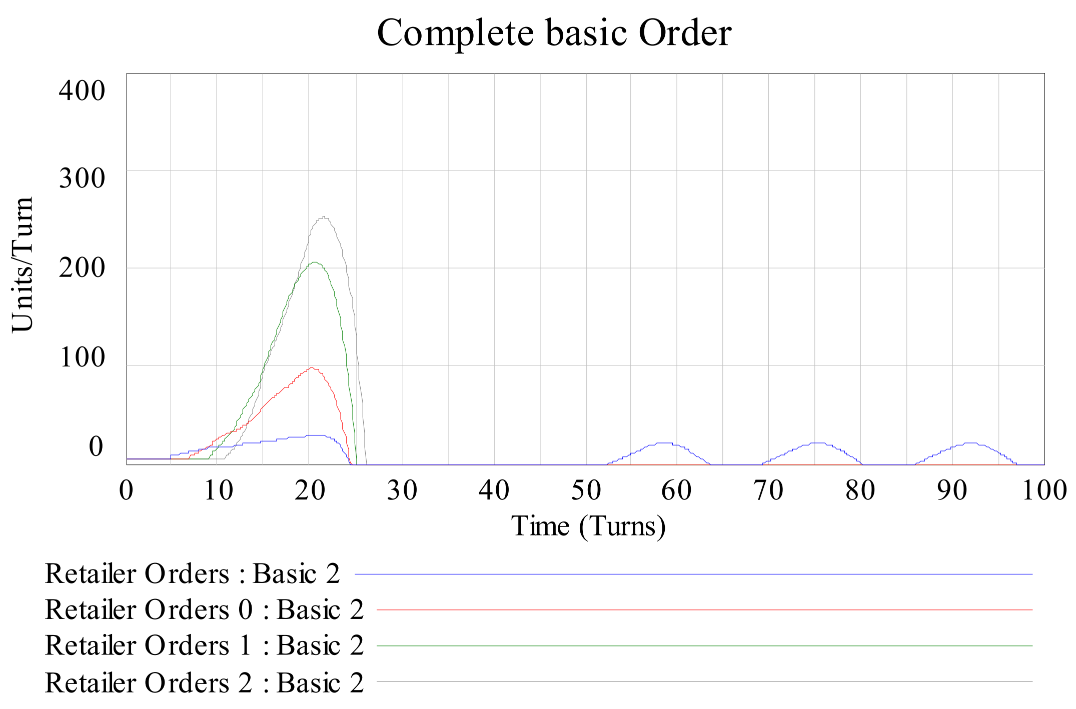


Illustration 14: Simple Ordering in a complete system

The Sterman Equations

The Sterman Beer Game, it is a classic model of normal supply chain dynamics. After it was initially created Sterman himself made a series of equations that defined the responses of a given individual during the circumstances of the Beer Game. These equations covered three key aspects of the way an individual might respond- adjustment for new order rates, response to insufficient inventory, and supply chain accounting. Most problems in the game, naturally, come from a poor account of the supply chain. Take, for example, the case of an individual who orders the amount that he needs to cover his backlog each and every time. This individual will have a high value (likely 1) for alpha, the variable defining response to insufficient inventory, and would have no change due to new ordering, and no accounting for the supply-line. This is the worst-case scenario for the game, as the individual will be slow to response to the rise in orders and will quickly order far too much in response to them. The result is a massive increase in the variability of order he produces, likely causing around 20 times the variation that the customer causes. Even if all other components in this case are responding ideally, the system will still behave terribly. Conversely, the ideal case is one where the individuals order what they expect to be needed in the future, accommodate for the supply line entirely, and order from inventory to correct for deficiencies in their predictions. When the supply line has more than they want, because they responded too vigorously, the individual would then order less than the predicted requirement. The result is a system that does not look at all like the normal beer game, and behaves very well.

When following the Sterman equations for the various steps in the supply line, a few things become apparent. The lower the beta, the accommodation for supply line, the worse it is to have a higher alpha, the response to inventory deficiencies. In fact, the relation of alpha and beta at the different stages causes almost all of the changes in overall behavior of the system. It is possible to change varying aspects with theta and the desired inventory settings, however they very minimally affect the shape. Changing the conditions of initial inventory and ordering processes in this model he has created also affect this equally little. If one was to alter the initial orders, while maintaining all other aspects of the system, the behavior changes incredibly little. Increase in variability at any given step only changes by about 1% per increase. The change is so minimal that in a Sterman-modeled system the same shape will persist even if it would cause the players to all have a backlog of over 100 for most the game. The reason for this is quite simple. The Sterman equations define a pattern of behavior, not a method of response. There is no development in these systems, nor is there any real attachment to the goals of the game. It can be responded that the changes in these equations are then not part of this written system, but rather changed based upon these initial parameters. Admittedly, the Sterman equations are definitely valid, they can model the system perfectly well within certain confines. Indeed, their own inability to change makes them especially well suited for how changes in policy for a group can alter the long term responses of the groups around them, causing either increased or decreased performance. In this way Sterman's equations are excellent representations of the interactions of static decision-making policies, however something more is required if the policies are to become more dynamic.

For reference, the equations written out are

Orders=Expected Orders+Alpha(Desired Inventory-Inventory-Beta*SupplyLine)*

*Expected Orders=Incoming Orders(t-1)*Theta+(1-Theta)*Expected Orders(t-1)*

SupplyLine=INTG(Orders-Incoming Shipments)

Alpha=Strength of Correction(0-1)

Beta=Accounting of Supply-Line(0-1)

Theta=Rate of Adjustment of Expectations(0-1)

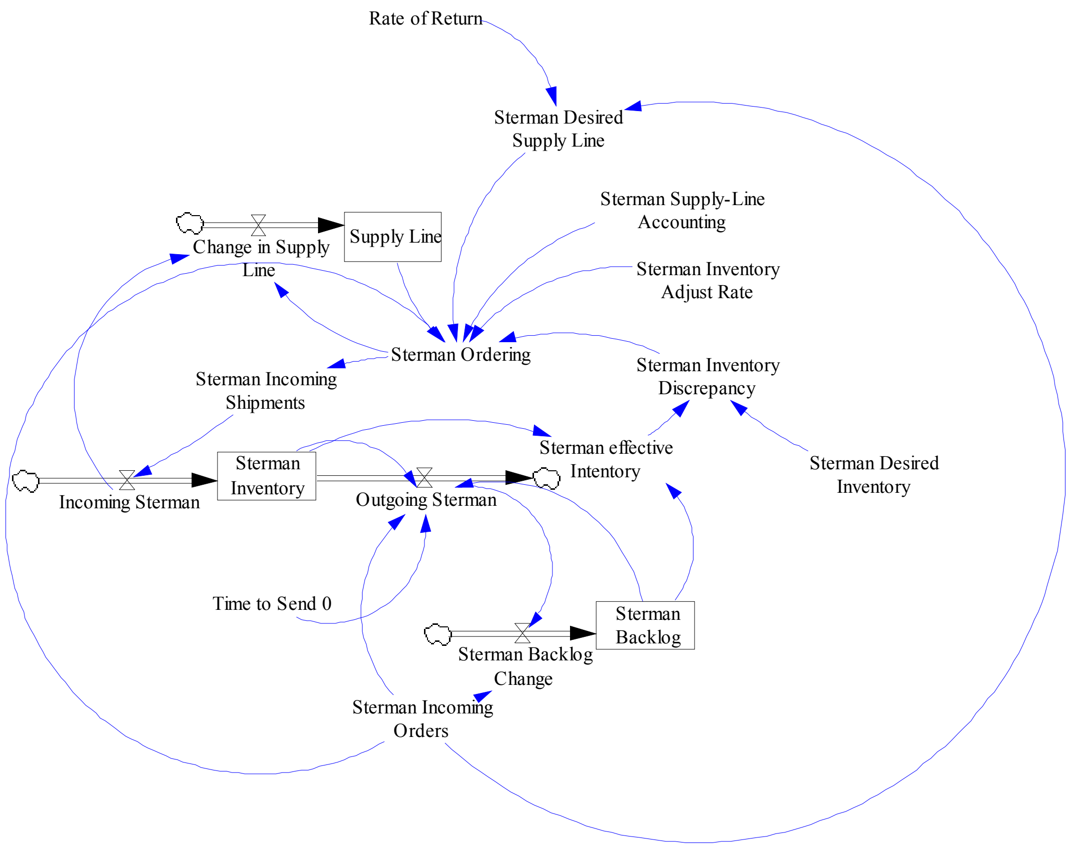

Rate of Return
Sterman Desired Supply Line
Sterman Supply-Line Accounting
Sterman Inventory Adjust Rate
Change in Supply Line
Supply Line
Sterman Ordering
Sterman Inventory Discrepancy
Sterman Incoming Shipments
Sterman effective Intentory
Sterman Desired Inventory
Sterman Inventory
Incoming Sterman
Outgoing Sterman
Time to Send 0
Sterman Backlog
Sterman Backlog Change
Sterman Incoming Orders

Illustration 15: Sterman Isolated Ordering Structure

When these equations are applied to a systems model of the Beer Game they take on relatively unremarkable properties. They produce an oscillating response that is dampened based on how much they account for the supply line. Take, for example, the case of an isolated Sterman decision-making structure where Alpha=1 and Beta=0.5. Generally this is an unrealistically large Alpha compared to those proposed by Sterman (Sterman, 1987), but it shows the general behavior his equations represent more clearly than smaller values might. As can be seen below the result is a shock followed by a weak oscillation. The values do not reach a particularly extreme level, tempered by a memory of the supply-chain and/or current inventory levels. This behavior is similar to that which can be seen during a multi-sector Beer Game model, but is significantly less extreme.

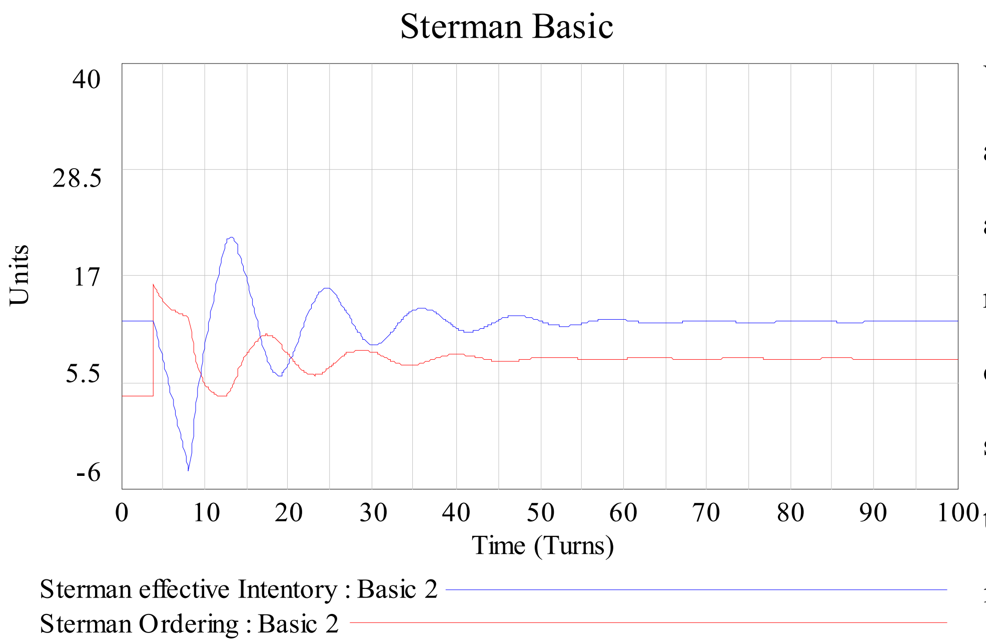


Illustration 16: Isolated Sterman Ordering - Behavior

When the the heuristic is applied to all four sectors of a Beer Game model, the result is the general behavior of the Beer Game. As can be seen in the below figure there is an unusual tendency in Sterman's equations for a steady-state to be reached with non-ideal conditions. When the desired supply-line is calibrated properly the result is an exceptionally well dampened system, even when there is only a moderate amount of supply-line accounting. This is only true if they do not have a strong reaction towards inventory correction. If there is a large value for Alpha (that is to say the inventory correction tendency) the behavior will be extremely poor. As can be seen in the below figure, the results are exceptionally poor. This is because the heuristic has nothing to prevent over-ordering if the supply line is being improperly accounted for and there is a strong response to correct inventory discrepancies.

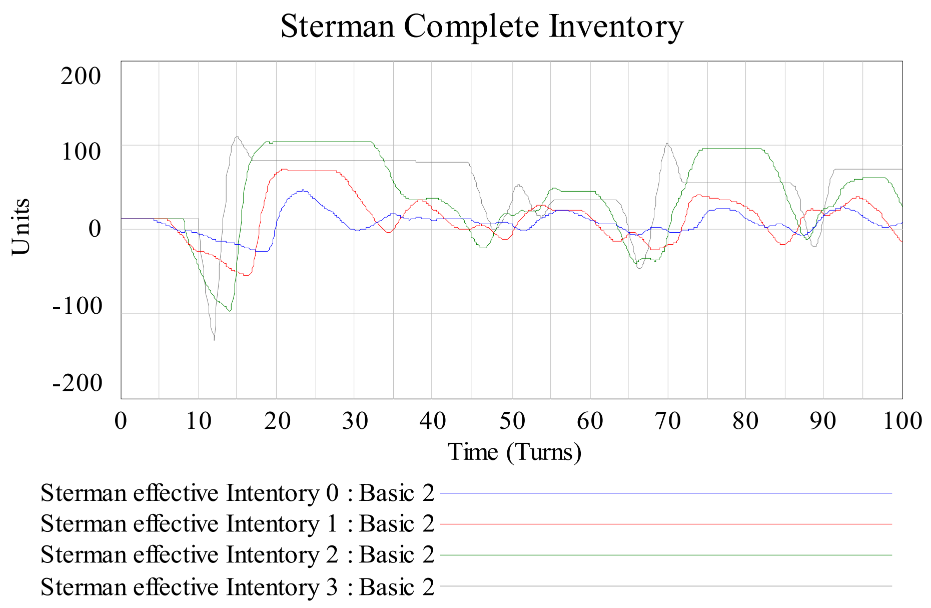


Illustration 17: Complete Sterman response with a high Alpha

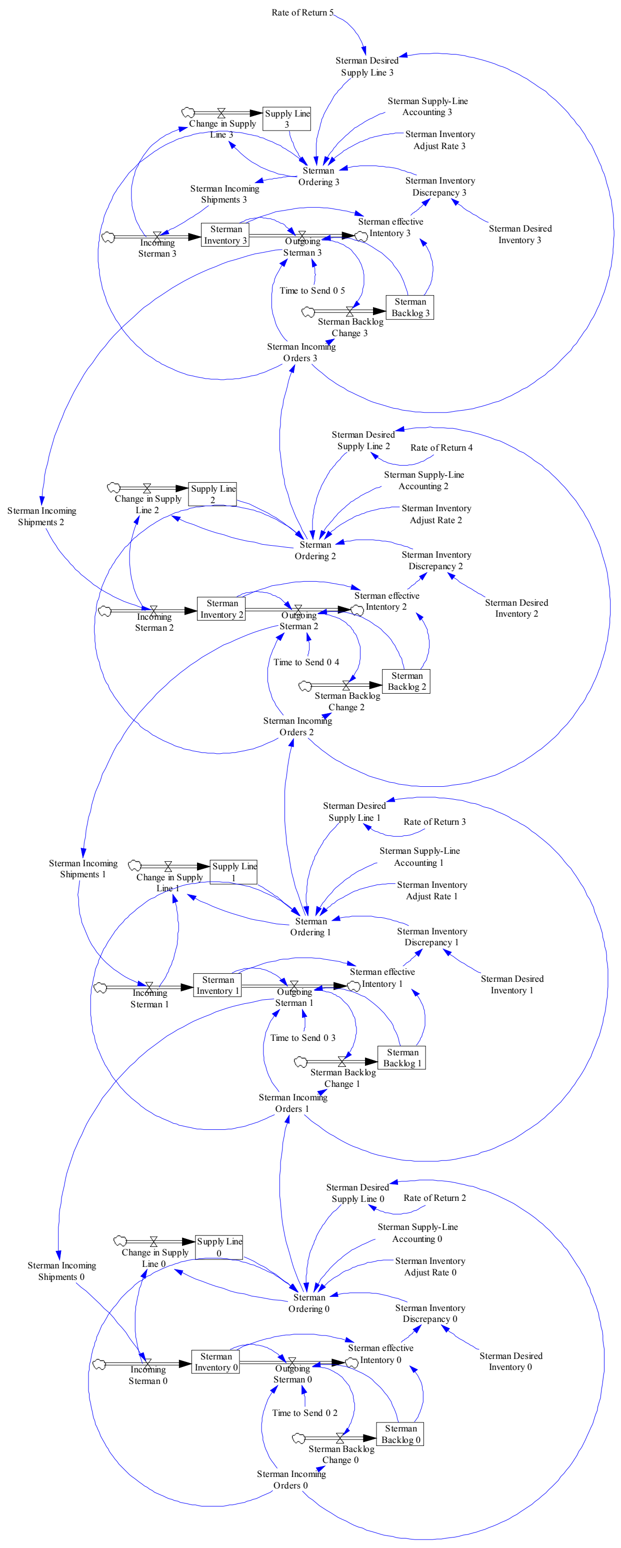

Rate of Return 5
Sterman Desired Supply Line 3
Sterman Supply-Line Accounting 3
Supply Line 3
Change in Supply Line 3
Sterman Inventory Adjust Rate 3
Sterman Ordering 3
Sterman Inventory Discrepancy 3
Sterman Incoming Shipments 3
Sterman Inventory 3
Incoming Sterman 3
Outgoing Sterman 3
Sterman effective Intentory 3
Sterman Desired Inventory 3
Time to Send 0 5
Sterman Backlog 3
Sterman Backlog Change 3
Sterman Incoming Orders 3
Sterman Desired Supply Line 2
Rate of Return 4
Sterman Supply-Line Accounting 2
Supply Line 2
Change in Supply Line 2
Sterman Incoming Shipments 2
Sterman Inventory Adjust Rate 2
Sterman Ordering 2
Sterman Inventory Discrepancy 2
Sterman Inventory 2
Incoming Sterman 2
Outgoing Sterman 2
Sterman effective Intentory 2
Sterman Desired Inventory 2
Time to Send 0 4
Sterman Backlog 2
Sterman Backlog Change 2
Sterman Incoming Orders 2
Sterman Desired Supply Line 1
Rate of Return 3
Sterman Supply-Line Accounting 1
Sterman Incoming Shipments 1
Supply Line 1
Change in Supply Line 1
Sterman Inventory Adjust Rate 1
Sterman Ordering 1
Sterman Inventory Discrepancy 1
Sterman Inventory 1
Incoming Sterman 1
Outgoing Sterman 1
Sterman effective Intentory 1
Sterman Desired Inventory 1
Time to Send 0 3
Sterman Backlog 1
Sterman Backlog Change 1
Sterman Incoming Orders 1
Sterman Desired Supply Line 0
Rate of Return 2
Sterman Supply-Line Accounting 0
Sterman Incoming Shipments 0
Supply Line 0
Change in Supply Line 0
Sterman Inventory Adjust Rate 0
Sterman Ordering 0
Sterman Inventory Discrepancy 0
Sterman Inventory 0
Incoming Sterman 0
Outgoing Sterman 0
Sterman effective Intentory 0
Sterman Desired Inventory 0
Time to Send 0 2
Sterman Backlog 0
Sterman Backlog Change 0
Sterman Incoming Orders 0

Illustration 18: Full Sterman Structure

Simplified Full Model – Kleinmuntz

There is an alteration of Sterman's original model which one should pay special attention to. That is the heuristic proposed by psychologist Kleinmuntz, which argues that players are not taking account of the supply-line, because that takes much more conscious effort than is likely occurring(Kleinmuntz, 1993). Instead, he proposes players taking into account only the immediate difference between their orders and the shipments that they are receiving.

There are several advantages to this model from a theoretical standpoint. It can produce behaviors very similar to the Sterman model, takes a great deal less processing power on the part of the player, and helps to provide some more explanation as to why the players make their decisions. That is not to say that is perfect. The bold section of the below graph represents the normal Sterman response in a situation while the dotted section is the response that his proposed equations would provide.

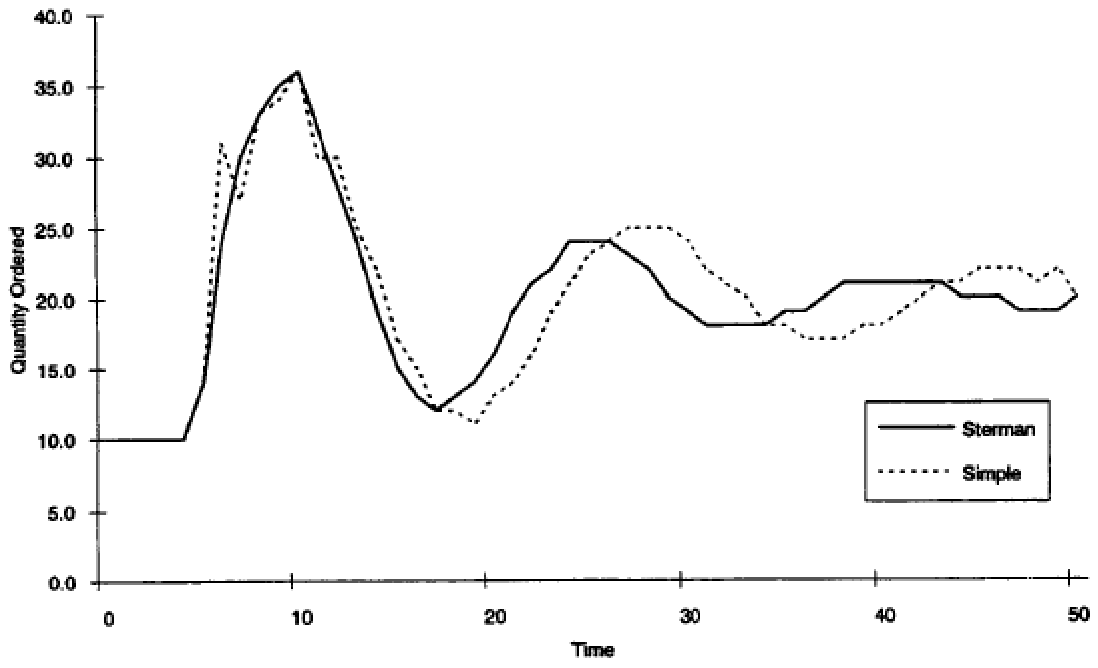


Illustration 19: Kleinmuntz's Original Basic-Structure Comparison

It is true in the highly simplified scenarios that Kleinmuntz first described that the behaviors between the two equations would be very similar. The above case represents a scenario of variables unlikely for humans aswell. When this test is recreated using approximately the same variables used, the behavior is very similar.

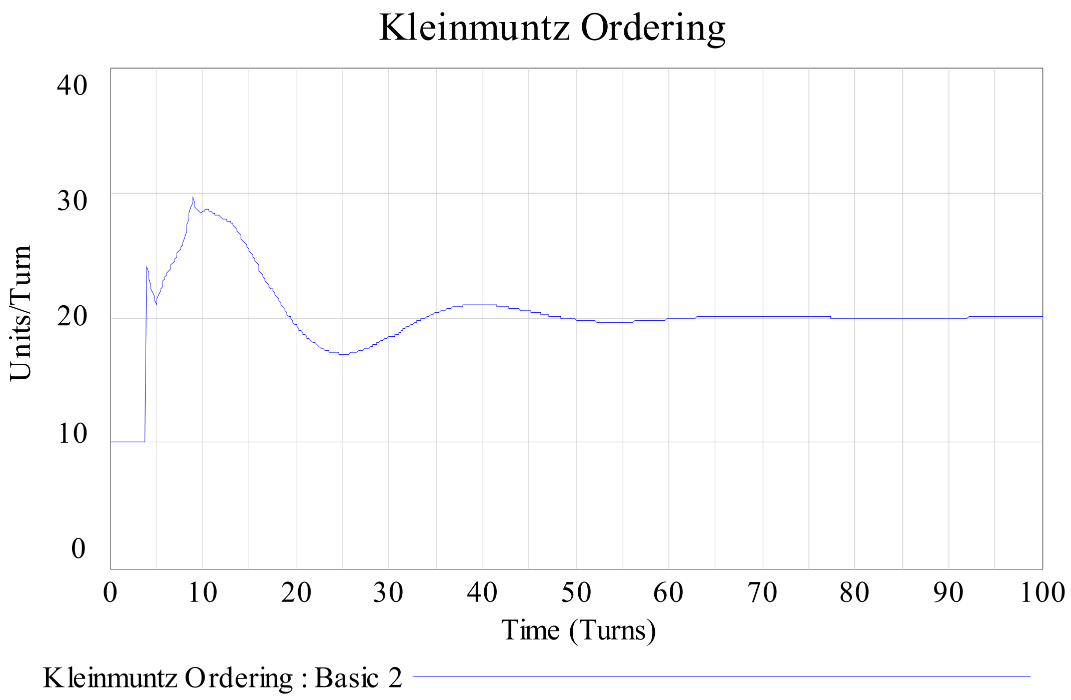


Illustration 20: Kleinmuntz Basic Isolated Orders

It should be noted that the base ordering for this behavior is 10, which doubles at time 4. This differs from the basic Sterman Beer Game, which starts with a demand of 4 and then doubles to 8 at time 4.

The result is slightly worse behavior that is highly correlated with the Sterman response. Again, as the situation grows more complicated, the simple version grows increasingly divergent.

Below is an example of the model outside the beer game model itself, where the sector both orders for itself and receives orders from the customer. (Kleinmuntz, 1993) The equations depicted are very similar to Sterman's and likewise are a more or less situationally calculated process. There is no buildup of concepts over time any more than the adjustment of perceptions of incoming orders.

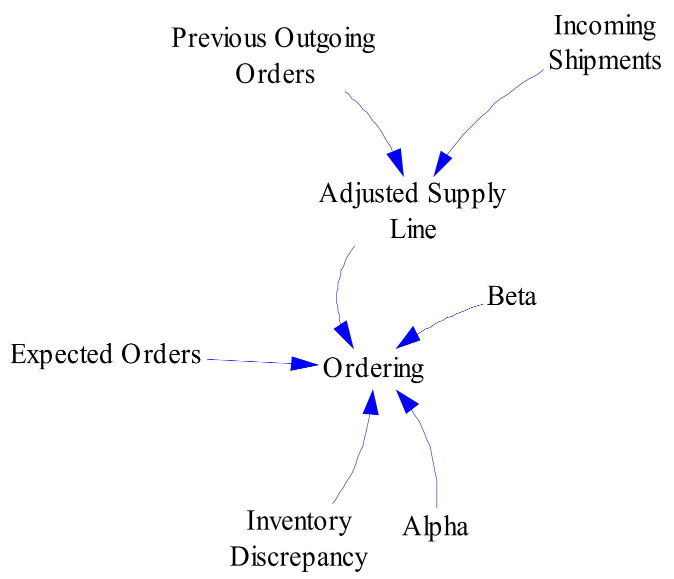


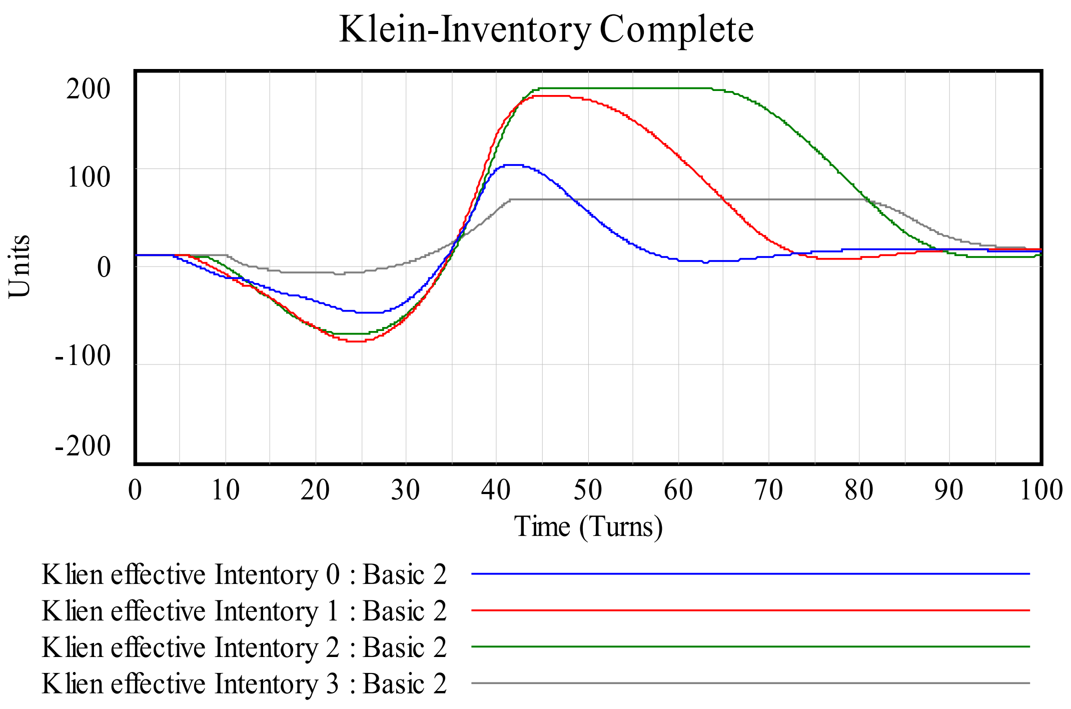
Illustration 21: Kleinmuntz Ordering Structure

*Ordering=Expected Orders+(Inventory Discrepancy-Adjusted Supply Line*Beta)*Alpha*

Adjusted Supply Line=Incoming Shipments-Ordering(t-1)

The only difference between this and Sterman's calculation is the way in which the supply line is accounted for. The model is incomplete because it is insufficiently robust. The supply chain accounting does not account for instances during which a player would have a clear knowledge that they have ordered too much, though such instances can be seen numerous times during actual game data.

When the structure is made into a full model, the behavior is generally poor. The reason for this is that while Sterman's model creates demands on the player which are unrealistic, Kleinmuntz simply does not expect enough from the players.. The below graphs are from a system that uses this decision-making heuristic exclusively. The problem is that the players have no memory of what they ordered in any way. If there is a delay of more than a few turns between ordering and receiving goods the players will have already ordered far more than was realistic. Kleinmuntz's model works well in isolation because the delays might reach up to four turns, but when it is expanded into a full system, there might



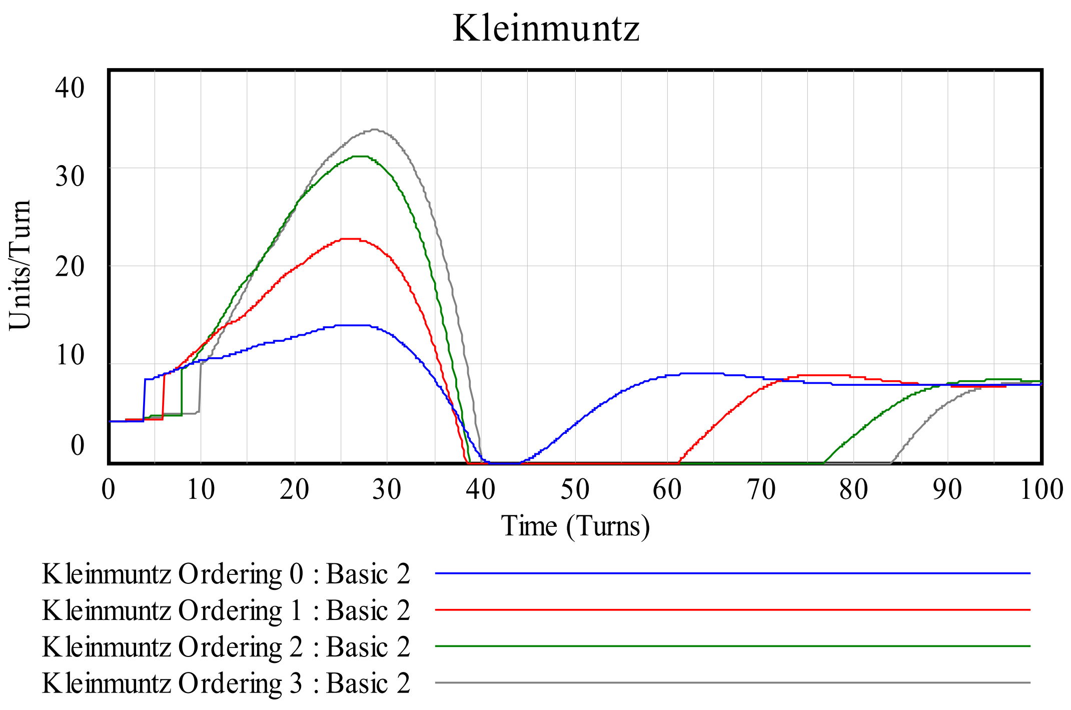
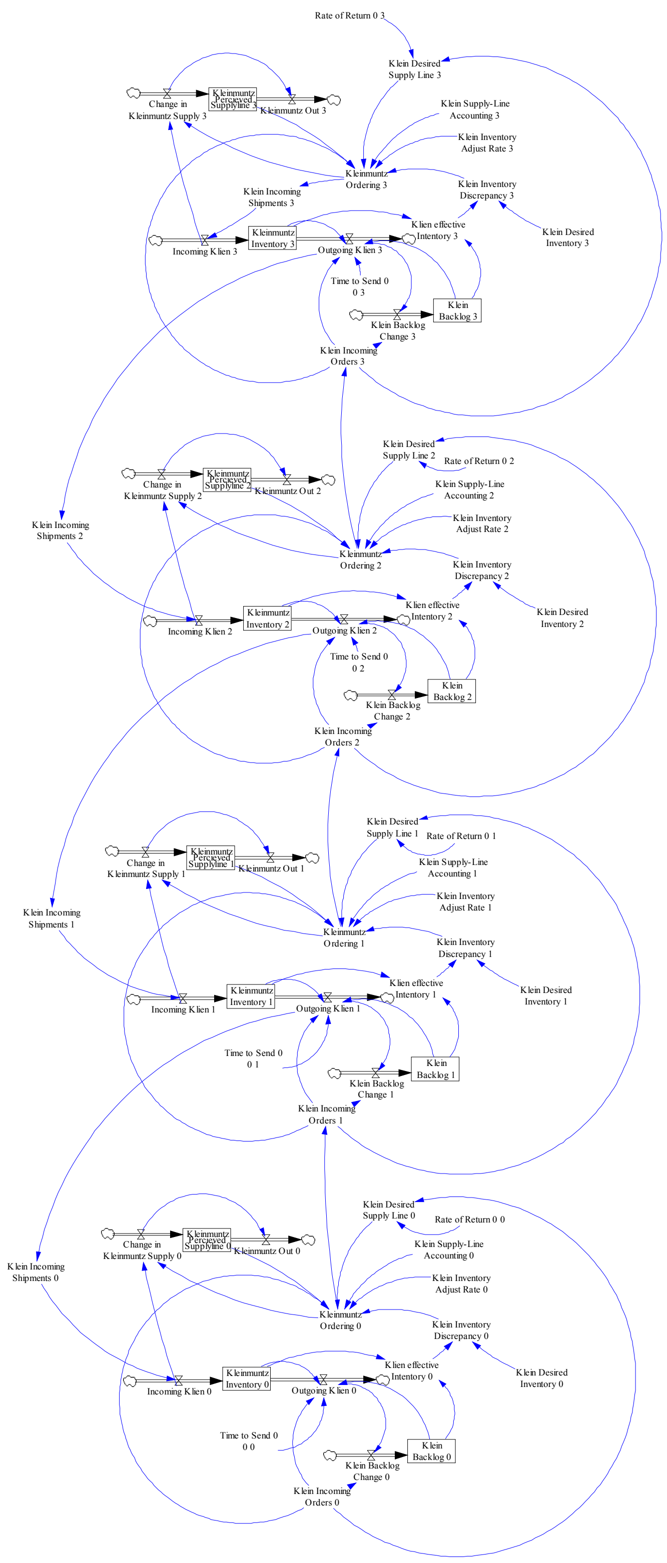


Illustration 22: Full Kleinmuntz System Behavior

be delays as high as 16 turns between ordering and receiving goods. The actions of the sectors during this time create huge problems that are quite simply unrealistic. Fortunately the ordering method does not include any steady-state errors, but simply generates massive problems that take a long time to



Rate of Return 0 3
Klein Desired Supply Line 3
Kleinmuntz Perceived Supplyline 3
Change in Kleinmuntz Supply 3
Kleinmuntz Out 3
Klein Supply-Line Accounting 3
Klein Inventory Adjust Rate 3
Kleinmuntz Ordering 3
Klein Incoming Shipments 3
Klein Inventory Discrepancy 3
Klien effective Intentory 3
Klein Desired Inventory 3
Kleinmuntz Inventory 3
Incoming Klien 3
Outgoing Klien 3
Time to Send 0 0 3
Klein Backlog 3
Klein Backlog Change 3
Klein Incoming Orders 3
Klein Desired Supply Line 2
Rate of Return 0 2
Kleinmuntz Perceived Supplyline 2
Change in Kleinmuntz Supply 2
Kleinmuntz Out 2
Klein Supply-Line Accounting 2
Klein Inventory Adjust Rate 2
Klein Incoming Shipments 2
Kleinmuntz Ordering 2
Klein Inventory Discrepancy 2
Klien effective Intentory 2
Klein Desired Inventory 2
Kleinmuntz Inventory 2
Incoming Klien 2
Outgoing Klien 2
Time to Send 0 0 2
Klein Backlog 2
Klein Backlog Change 2
Klein Incoming Orders 2
Klein Desired Supply Line 1
Rate of Return 0 1
Kleinmuntz Percieved Supplyline 1
Change in Kleinmuntz Supply 1
Kleinmuntz Out 1
Klein Supply-Line Accounting 1
Klein Inventory Adjust Rate 1
Klein Incoming Shipments 1
Kleinmuntz Ordering 1
Klein Inventory Discrepancy 1
Klien effective Intentory 1
Klein Desired Inventory 1
Kleinmuntz Inventory 1
Incoming Klien 1
Outgoing Klien 1
Time to Send 0 0 1
Klein Backlog 1
Klein Backlog Change 1
Klein Incoming Orders 1
Klein Desired Supply Line 0
Rate of Return 0 0
Kleinmuntz Percieved Supplyline 0
Change in Kleinmuntz Supply 0
Kleinmuntz Out 0
Klein Supply-Line Accounting 0
Klein Inventory Adjust Rate 0
Klein Incoming Shipments 0
Kleinmuntz Ordering 0
Klein Inventory Discrepancy 0
Klien effective Intentory 0
Klein Desired Inventory 0
Kleinmuntz Inventory 0
Incoming Klien 0
Outgoing Klien 0
Time to Send 0 0 0
Klein Backlog 0
Klein Backlog Change 0
Klein Incoming Orders 0

Illustration 23: Full Kleinmutz Structure

correct. It should be noted that the behavior is not always this poor. The model can reflect realistic situations more closely, as well as Sterman's equations. This simply reflects the most common output this method will deliver. Additionally, the method isn't sufficiently robust that it can deal with problems. If even one sector behaves poorly, this is the sort of result that will occur.

As far as the change in variability is concerned, there is little difference between this concept and the Sterman equations. The key differences between the two is that this one is more prone to variability at a lower level, however this might simply be a result of attempting to recreate the results of the Sterman equations rather than actual game data. When used to recreate a more simplified version of the Beer Game, the results were quite good(Kleinmuntz, 1993). The reason for the lack of accuracy in this example may simply be because of insufficient matching techniques used. The two models are not directly equatable, different parameters must be used when approximating the weight of the supply-line. A Sterman model that fully accounts the supply-line, for example, is vastly different than simply fully accounting the momentary supply-line of the simple model.

Overall it should be restated that the original differences proposed by Kleinmuntz have shown to be situationally unlikely. That is to say, there are certain situations present where they generate improbable results. When the model has players receiving significantly more than they have ordered, it will create an unusual tendency to heavily under-order. In a normal pattern this can be a good thing, as the cycling patterns of response-strength make the ordering more responsive. In the case of a more complex system, though, players can easily under-order for these reasons. The result create strong under-ordering in a number of situations where the player still clearly needed additional stock. If the structure is given a strong response, then over-ordering is just as likely an outcome. Additionally, when one looks at the retailer sector of actual game data, correlation is found to be very weak. The correlation when calculated by PMCC was determined to be highly variable, anywhere between 0 and . 56 in actual data. In general it was very small (.06-.12), though there was no consistent way to determine the response. This data was only taken from the retailer sector of actual games, therefore cannot be generalized entirely. Just a general glance at the data will reveal frequent events of players continuing to order large quantities while deliveries exceed orders by wide margin. Oftentimes this does not even cause significant over-order because the player has properly accounted for the supply chain(or perhaps has coincidentally stopped ordering at the right time). It is possible that the correlation is non-linear, and the players tend to account over-response differently than under-response when ordering, however no such correlation was tested during these simulations.

Better Safe Than Sorry

This is a heuristic theorized to explain the behavior of players during simulation-games other than the Beer Game. The basic concept is that players will respond to the time-constants of the given situation by giving a response that ensures at least the minimum necessary response, though they might not actually process what the ideal response itself is. Originally this was meant to deal mostly with a fire-fighting simulation by Brehmer and Nählinder (2004) as well as later by Elg(Brehmer, 2005). The specifics of this concept have been adjusted for the purposes of the Beer Game.

For the purposes of the Beer Game this structure forms one of the more complicated Systems that will be presented today, but it is inherently a simple processes for the mind to perform.
The basic way the structure performs is conceptually simple. Like in a simple ordering system, the

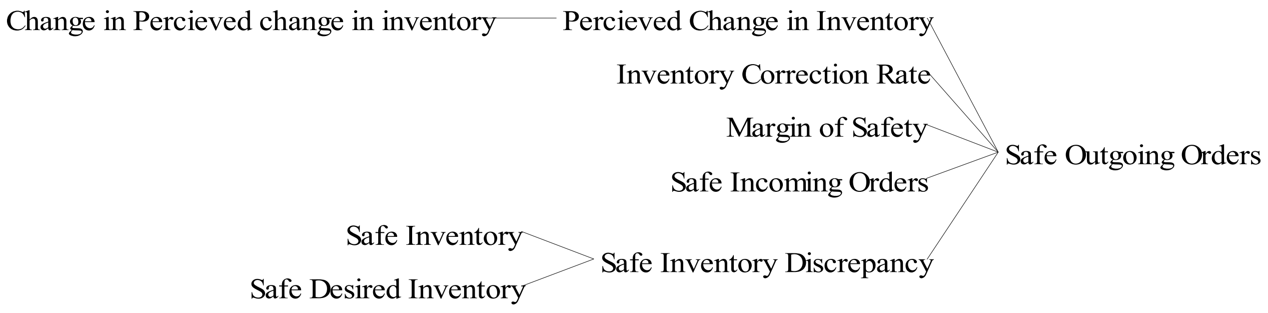

Change in Percieved change in inventory
Percieved Change in Inventory
Inventory Correction Rate
Margin of Safety
Safe Incoming Orders
Safe Outgoing Orders
Safe Inventory
Safe Desired Inventory
Safe Inventory Discrepancy

Illustration 24: Basic Safe Ordering Patterns
incoming orders form the base of the ordering process. This value is increased or decreased by the difference in inventory and desired inventory multiplied by the inventory correction rate. Additionally, there is a 'safety response' based on the Perceived change in inventory. If the inventory is dropping very quickly, orders will increase, likewise, if inventory is increasing very quickly orders will quickly stop.

Safe Outgoing Orders=
MAX(0, Safe Incoming Orders-Margin of Safety*Perceived Change in Inventory-
Safe Inventory Discrepancy*Inventory Correction Rate)

Visually the basic structure seems more complicated because of the need to track perceived rate of change of inventory.

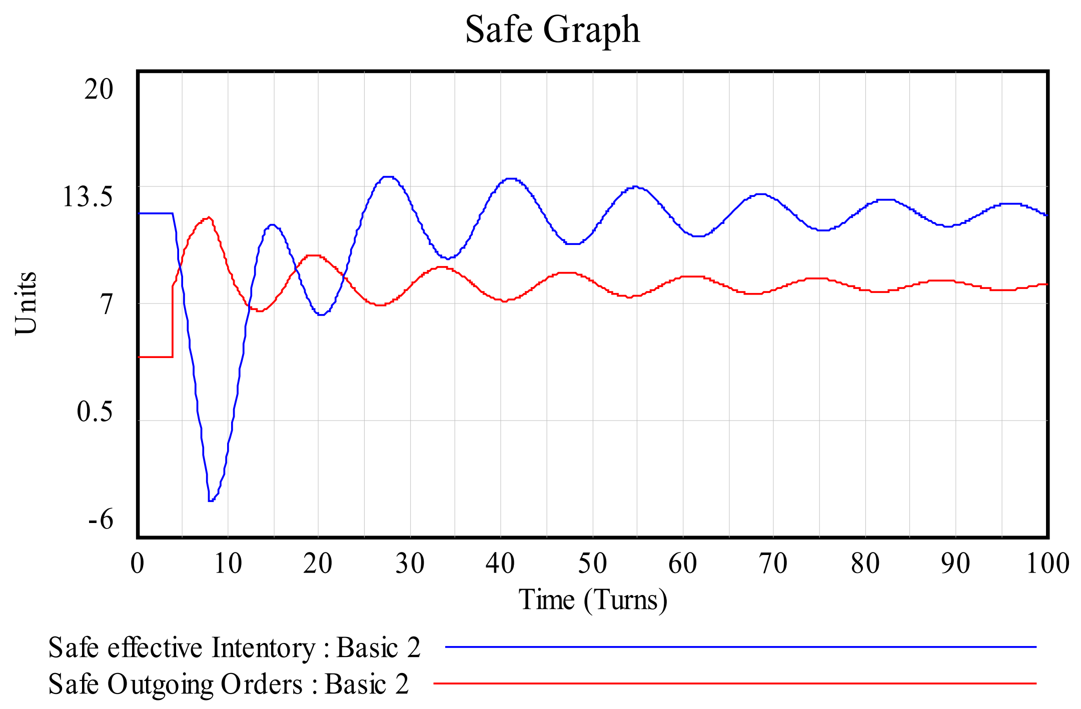


Illustration 25: Better-Safe-than-Sorry isolated behavior

In isolation this structure does not behave very poorly. It has a reasonable degree of oscillation, but it general it will quickly reach the ideal ordering response, and will not significantly over-order at any point. It is a tremendous improvement over the simple ordering methods. Interestingly, the structure develops a unique behavior when extrapolated into the full size of

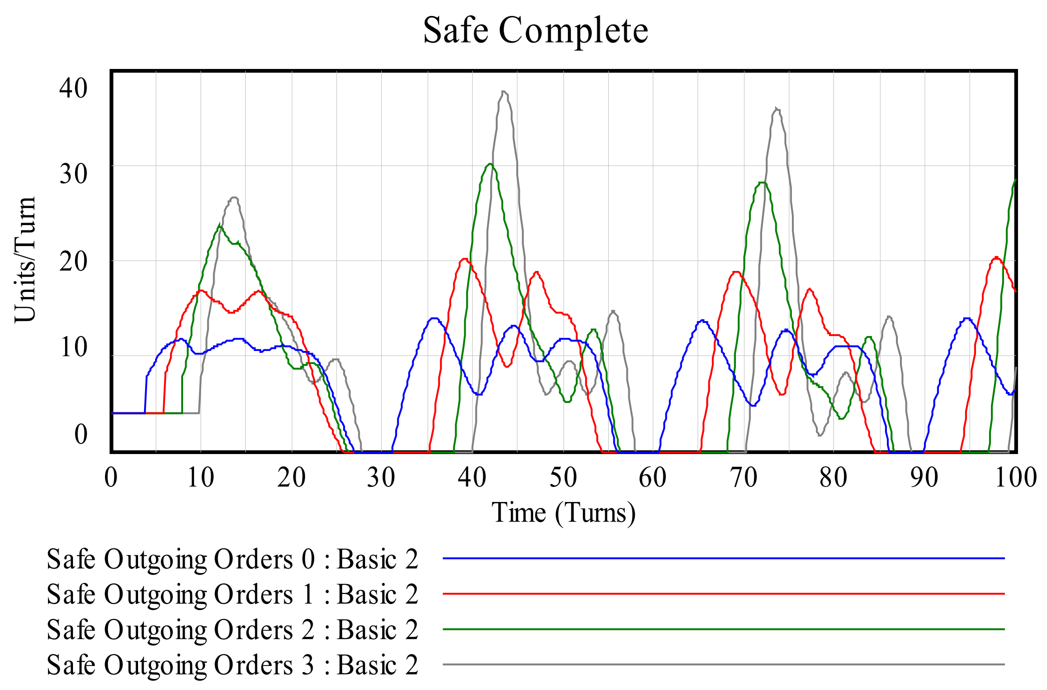


Illustration 26: Better Safe than Sorry – complete ordering behavior

the Beer Game.

The above figure 26 shows the responses generated by by this structure during standard conditions.

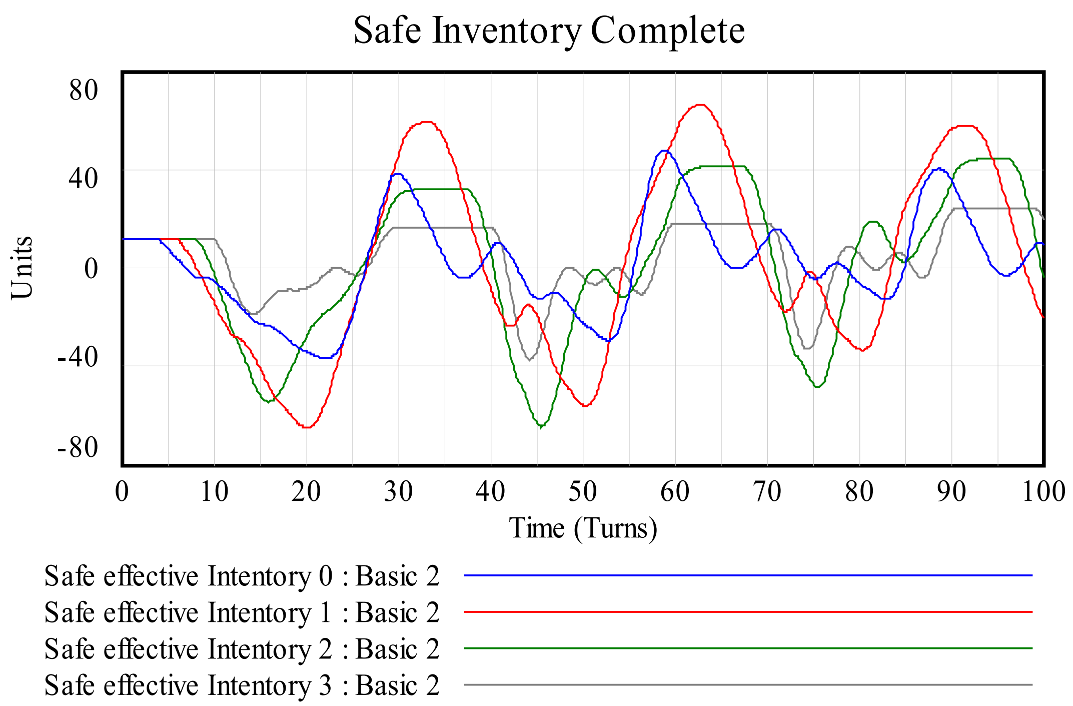


Illustration 27: Better-Safe-than-Sorry - complete inventory behavior

The above data is not type fit to any specifics individuals, but can be seen to be very similar to many game runs performed by human players. The single greatest failing is that these agents will not 'learn' in the proper sense, their reactions will always be as extreme and it is extremely unlikely they will ever reach a steady state. In the above example it would be hundreds of turns before the decision making structures reached equilibrium, something that would likely occur much earlier for human players. Though it is imperfect, this represents a fascinating and very possible heuristic that humans may well be using.

.

Anchor-Adjust

It is a well known phenomena that individuals will grow attached to a basic or key value they use during decision-making(Tversky, 1974). Decisions are often made relative to this basic value, a high initial value leading to a higher estimate, an initial problem leading to greater assumptions of issues in the future. In order to apply this concept to the Beer Game, a simple decision-making heuristic was used, and the decisions took time to adjust to this projected value. Therefore, if the system was already over-ordering, it might continue to do so after it had already become a bad idea.

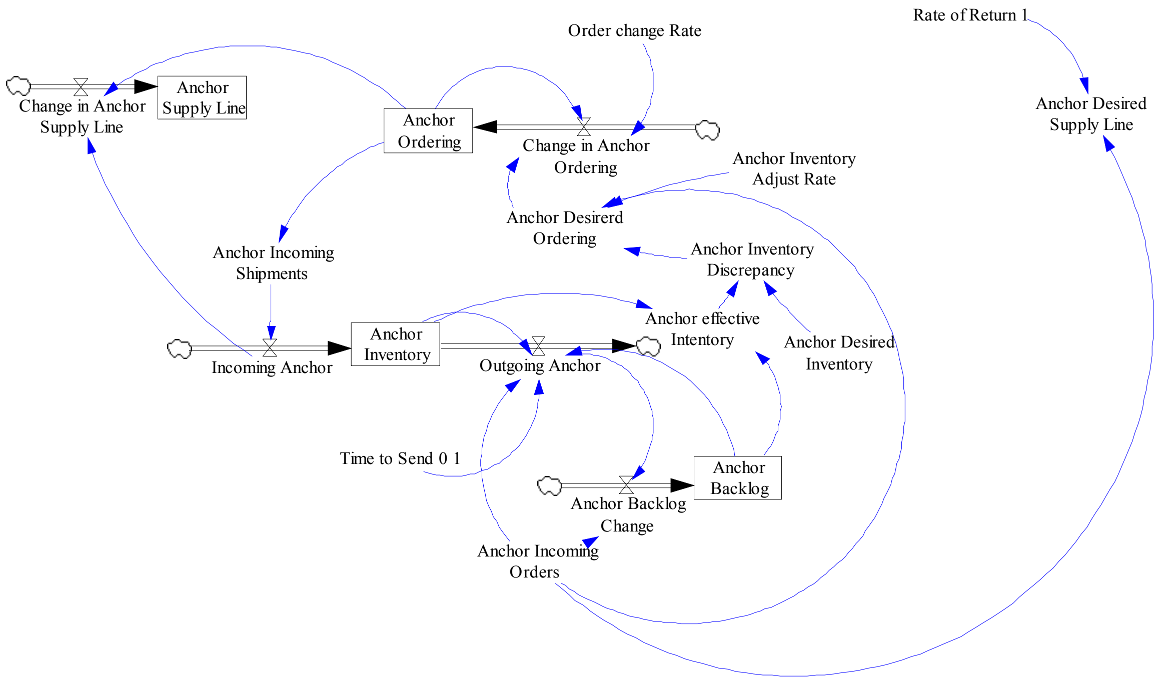


Illustration 28: Basic Anchor Ordering Structure

For basic testing the ordering had no inherent qualities that were different from the Simple-Ordering method, besides the existence of policy resistance. The result was incredibly bad behavior that in no way could explain that performed by normal humans during the beer game. While the concept may not be inherently wrong, it is clearly not performed on a personal basis with a method as simple as a basic ordering technique.

As can be seen below, the behavior is horrific. Over-ordering is in excess of 100 units even when the system is isolated. When the system is expanded to cover four sectors, ordering can exceed 3000 units a turn, because of the nature of the exponential growth the system causes, combined with the increased responses caused by this version of the Anchoring method.

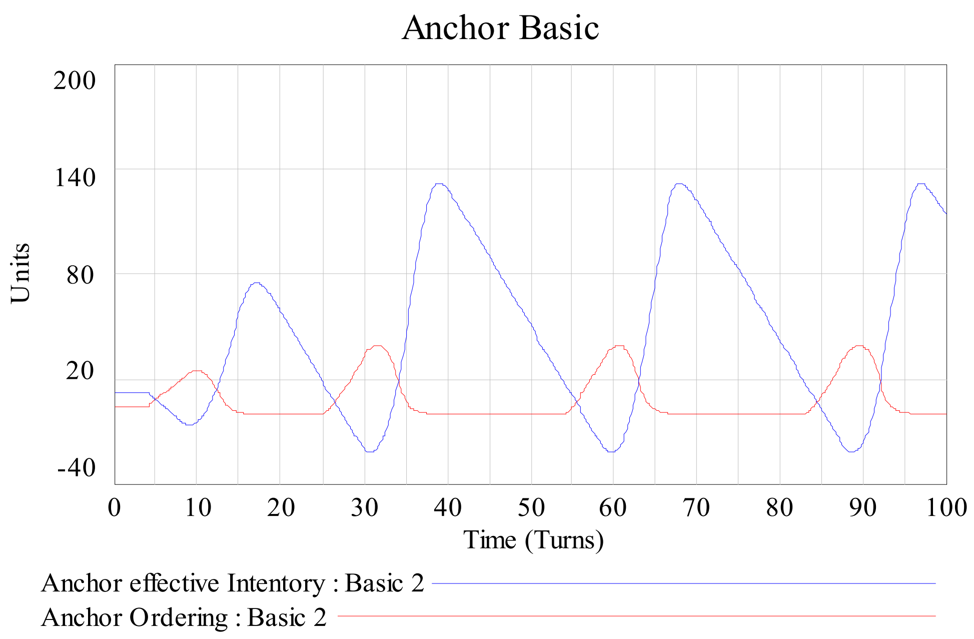


Illustration 29: Isolated Anchor-Behavior

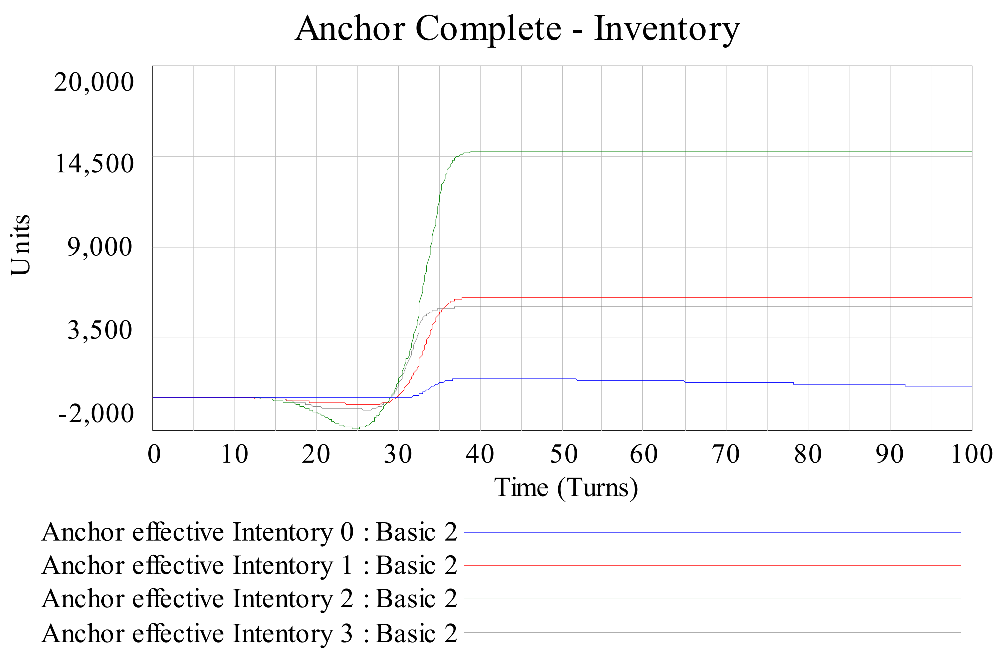


Illustration 30: Complete Anchor-Inventory Behavior

Unified Model

Each of these separate heuristics has yielded interesting behavior. While not all behaviors were ideal, many contained concepts that might explain phenomena that occur when players are making their decisions. The final approach that was attempted was the unification of the concepts that have already been expressed. There have been no significant attempts in creating a unified heuristic model, therefore there are no aspects of this that have been researched yet.

The foundation of the ordering structure is the Anchor-adjust system. Though it reached

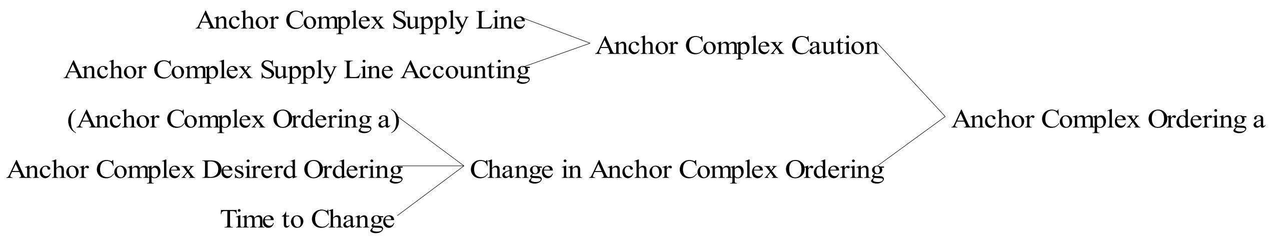

Anchor Complex Supply Line
Anchor Complex Supply Line Accounting
Anchor Complex Caution
(Anchor Complex Ordering a)
Anchor Complex Desirerd Ordering
Time to Change
Change in Anchor Complex Ordering
Anchor Complex Ordering a

Illustration 31: Complex Ordering Decision Process

extremely poor behavior on its own, it gives a base where the player's thoughts can be expressed. Rather than a specific calculation that is done at each moment it is an impression that changes over time depending upon the momentary circumstances. The Anchor-Adjust method is still very similar to the previous anchor-adjust model. The ordering response is determined by a goal-seeking behavior. Each turn the player determines what their goal should be based on a basic decision-making technique similar to what Sterman and Kleinmuntz described. Additionally, there is a cautionary response in the ordering habits of individuals. The 'supply line' individuals remember in this model is not a traditional recollection of how many products are going to be received, but instead a gradually diminishing memory of how much more or less the player has ordered in the past turns relative to the incoming orders. It is my conjecture that people account for the supply line in this way if at all – by remembering their own under or over-ordering practices, not trying to account for the very large delay that might occur.

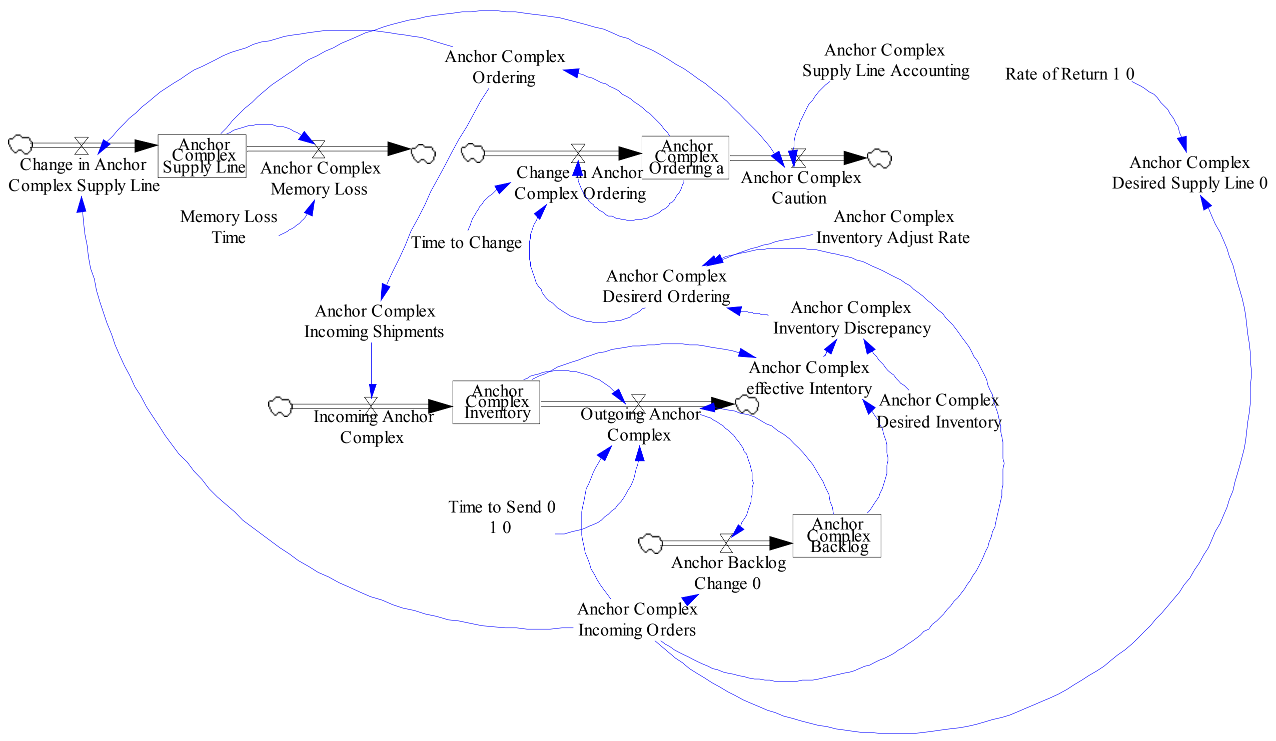

Anchor Complex Ordering
Anchor Complex Supply Line Accounting
Rate of Return 1 0
Anchor Complex Supply Line
Anchor Complex Memory Loss
Change in Anchor Complex Supply Line
Memory Loss Time
Change in Anchor Complex Ordering
Anchor Complex Ordering a
Anchor Complex Caution
Anchor Complex Desired Supply Line 0
Time to Change
Anchor Complex Inventory Adjust Rate
Anchor Complex Desirerd Ordering
Anchor Complex Inventory Discrepancy
Anchor Complex Incoming Shipments
Anchor Complex effective Intentory
Anchor Complex Inventory
Incoming Anchor Complex
Outgoing Anchor Complex
Anchor Complex Desired Inventory
Time to Send 0 1 0
Anchor Complex Backlog
Anchor Backlog Change 0
Anchor Complex Incoming Orders

Illustration 33: Unified Model Structure

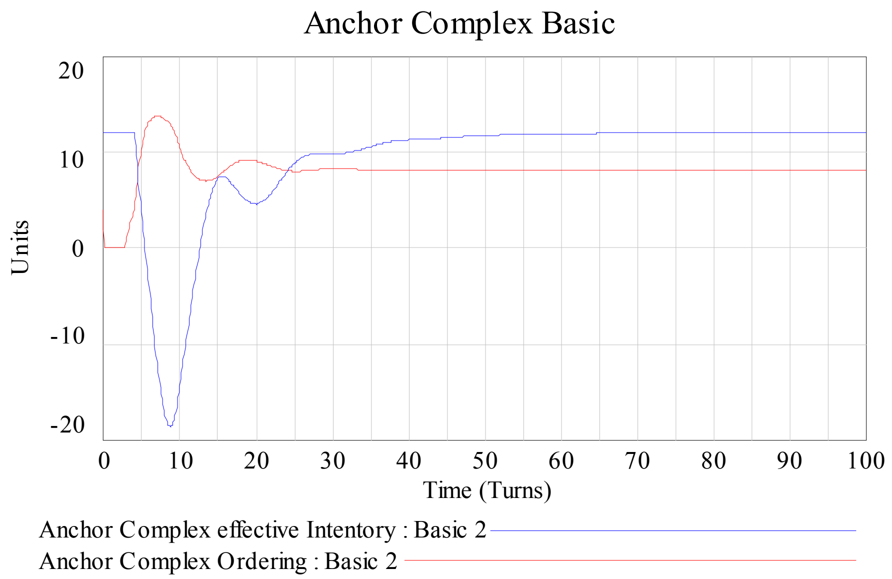
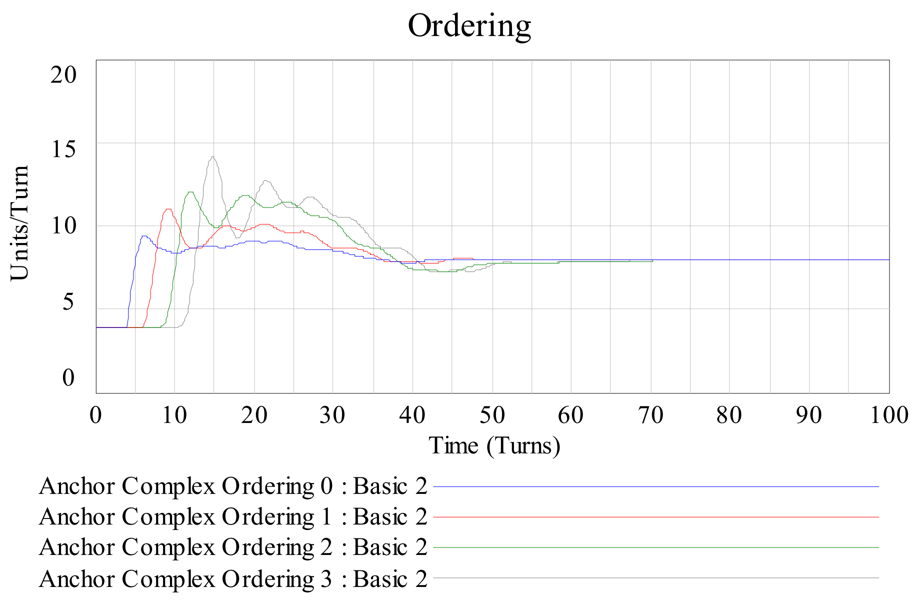


Illustration 32: Unified Model - Isolated Behavior

When placed in an isolated system, the decision-making structure operates fairly well. The result is a very effective goal-seeking behavior that will over-order goods when demand increases then quickly reduce ordering to a more reasonable level. Unlike many of the other structures, this will not achieve steady-state error during most situations. Instead the inventory levels will quickly reach the desired levels. When this structure is expanded out to a larger scale it creates a very interesting set of responses.

As can be seen in the below figure, the result of the simulation is very similar to those that humans output. There is a multiplicative increase in order variability from sector to sector, but the final sector (the 'factory' sector) still maintains a relatively consistent inventory level.



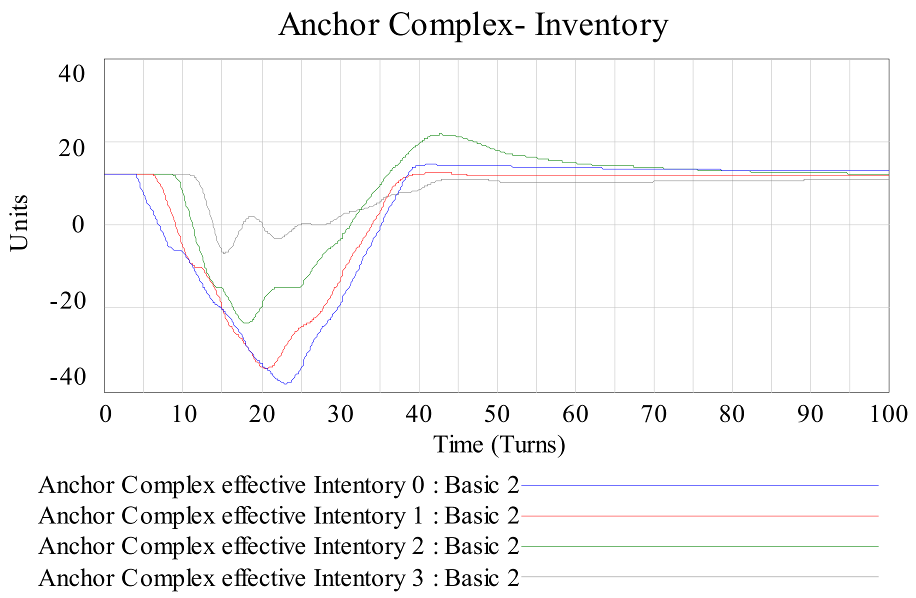


Illustration 34: Unified Model- Complete Behavior

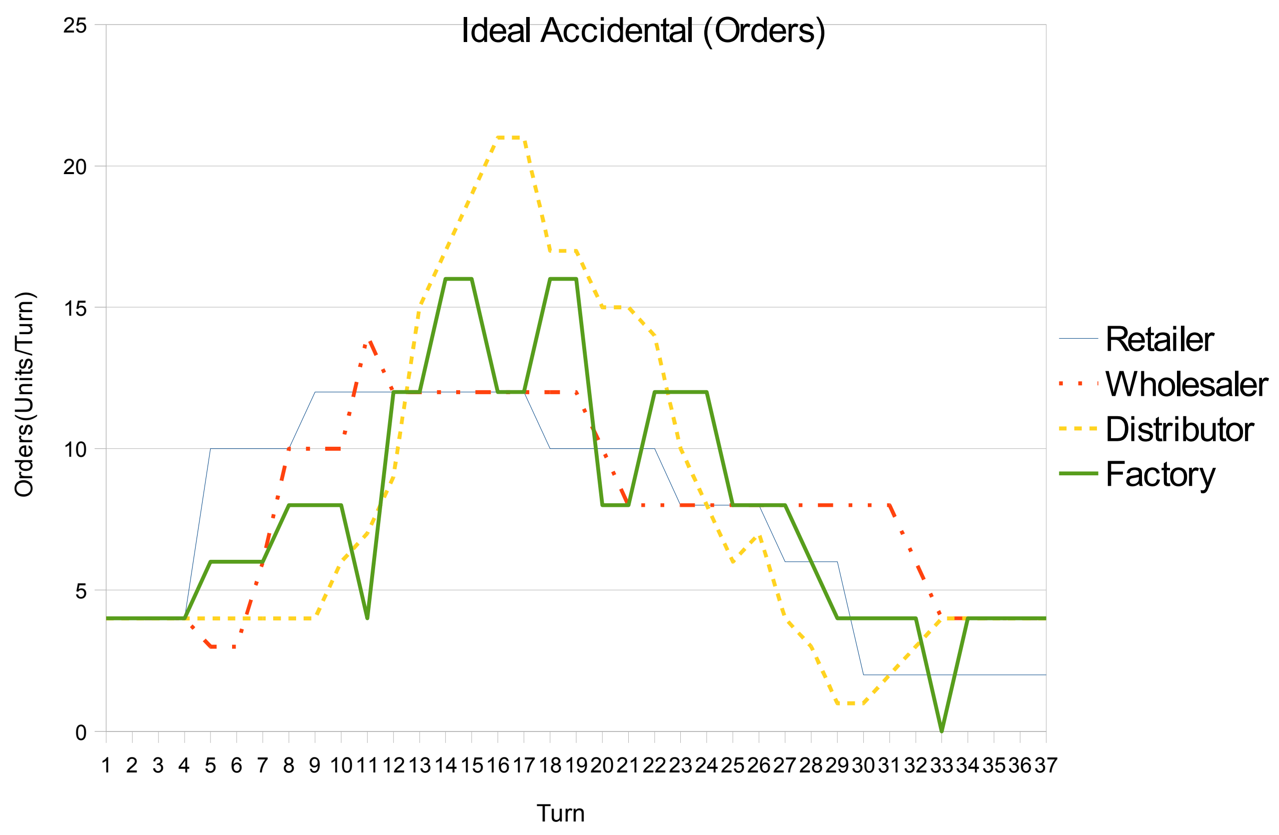
Analysis, , Towards a future model

The Sterman model is incredibly comprehensive and covers almost every range out outputs one would want. That said, it is unrealistic to assume that the model is a perfect representation of what occurs in the deliberation process for the beer game. All simulations are imperfect, but the Sterman model fails to include everything that would realistically occur in an individual or account for discrepancies with reality in some situations. The reasons for this are many-fold and have been discussed earlier. More than likely full human reasoning would be an adjustment between multiple different heuristic styles, such as that discussed in Kleinmuntz's earlier works. No single basic heuristic styling occurs, but instead players 'learn' and adjust based upon success to different but relatively simple systems.

What this does include, with significant implications, is that the capacity for a supply chain model to have shifting variance increase between the groups as conditions change, especially those conditions that make managing the model more difficult for the participants. It has been shown that an independent and slow to adjust modifier outside the normal decisions making process will create this change, as well as other more complicated systems that adjust independently of the actual state of the system Adjusting with perceived success or failure adds a dynamic that was previously missing in the Sterman model, and can give us important insights into the basics of supply chain dynamics. Most significantly, the bull-whip effect can 'disappear' in certain circumstances simply because the condition was too easy or the bias' of the individual happened to counteract the effects in the situation

Take for example the below graph, taken from an actual run of the beer game. Because of the bias of both the factory player and the distributor the results were excellent, abnormally so. They each ordered significantly more than they needed to as an initial buffer to deal with uncertainty, the factory doing this immediately and the distributor as a response to increased demand. The unusual bias of the players involved happened to compensate for what normally would be very poor behaviors. There was a dramatic response to failure and change in responses from their supplier, but there was no clear calculation of a supply chain, no exact cut-offs or precise measurements, just reserved ordering patterns which would indicate a knowledge of the presence of a supply lag.

There are a number of interesting results that these simulations show, besides the possibility of a new



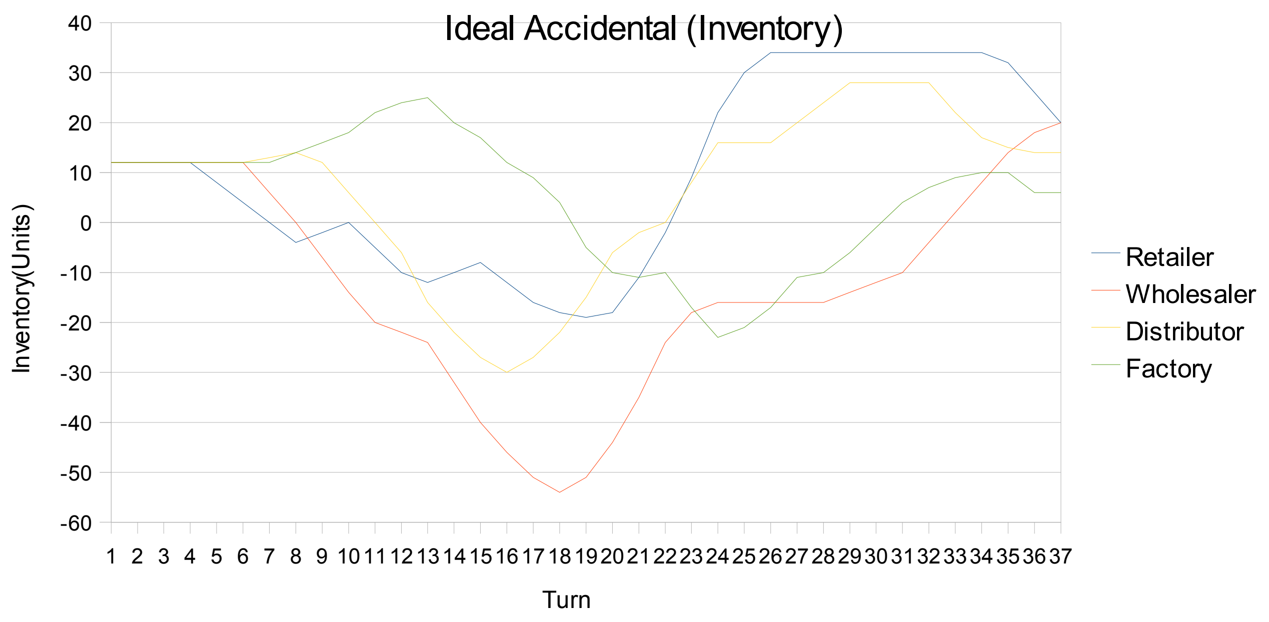


Illustration 35: Ideal Accidental Inventory/Orders

outlook on the modeling of the problem. Among the results is an interesting display of the reactions that the different sectors have to increases in ordering size with a consistent inventory level. Functionally, in the model this is just decreasing the relative inventory, though there might be some response on the part of the players because of the difference in the scale of numbers. What is significant about this discovery is that the sectors respond very differently to this change, their net variability has very different implications depending upon the setting. The retailer, for example, always escalates the net variability based upon the size of the orderings. The factory, however, has the exact opposite response. When the ordering size increased, the net variability of the factory and distributor consistently dropped. There are a number of potential reasons for this, but the as the delay to the end of the supply like decreases, so does the net variability. This implies that it is actually the responses of those above which determine a sector's variability, not its own. It is an interesting possibility that has a large number of implications. Most significantly, this would mean that when looking at a Supply-Chain, in a simulated Beer Game run or in reality, the supplier is what would dictate a sector's behavior, not that sector's own wishes. This would not be strictly true, but it would be true with regards to the variability of order. Those companies with erratic or unreliable suppliers would develop very poor ordering trends because of the lack of proper feedback. The actually implications for what occurs vary slightly depending upon which decision making heuristic they are found to be using.

There is some variation between the different models in this regard. For example, the escalation model predicts a fairly smooth increase in net variability for each level. This increase is less noticeable as it progresses from level to level, however this is a standard response for the model in general. The basic anchors and adjust model predicts a rapid rise as conditions become non-ideal(as it becomes less and less realistic to perform well), then a relatively slow increase after that point. The Sterman model predicts the lowest total change. Even when transitions are less smooth than during the escalation model, but follow the same general patterns. The above example is taken from a 'normal' example run of the Sterman game, but the specifics of differences in variation can change based upon settings.

Distributor and factory always have a decrease in net variability as time goes on for realistic parameters.

Additional experimental testing in this area could determine which of these models best explains the behavior of players during the beer game. It seems possible that different players could be best represented by different heuristic structures based upon the complexity of their decision making and commitment to the game. The discovered differences in net variation change could be used as a means of determining the specifics of how players make decisions during the beer game. This would be advantageous because players are notoriously unreliable in their described decision methods. The implications outside of the model itself are unusual as well. This implies that the unreliability of orders a company receives mean that it is less and less likely that their own ordering policies will have a significant impact. The reason that the factory and distributor have a low net variability that doesn't change significantly is largely because they are already interacting with orders of a very high variability. There is little additional fluctuation that these sectors can do while staying inside reasonable bounds. Likewise they are unable to predict or manage the behavior of the groups below them leading to a relatively confined set of responses. Because of their generally more constrained responses the factory and distributor tend to cause reduced net variability in systems where the bull-whip effect is strong.

In creating a final model to demonstrate everything that has been shown here, it would be important to take into account the wide variety of elements shown to exists within both the previously formulated models as well as the background research on the subjects of human performance in simulations and memory. This model would have several structures, each of which is found in humans in a complicated decision making environment.

First and foremost is the existence of the human tendency to take a safe-approximate route in their avoidance of time constant variables. The time constants are avoided because their short-term implications are extremely difficult to process without active attempts to remember. Current studies have shown that without intentional recollection of items, the realistic maximum number of 'chunks' a human can remember in the short term is approximately 4. Each of these 'chunks' is a group of closely related information that may or may not include specific values.(Cowan, 2000) The less specific the information, the more that can be held, leading to a great deal of difficulty when calculating the effects of this memory in a simulation. This is lower than the generally accepted seven plus or minus two. The reason for this discrepancy is that each chunk can contain multiple pieces of related information. During basic tests to remember items, it was found that when the recording process was subconscious, and there was no chance to group the information, significantly less information could be stored. Instead, only these four large 'chunks' of information could be retained.

This memory is not held or received entirely on a conscious level, but instead a combination of conscious and unconscious perceptions and processes help guide the player's decision making. This is what was meant by Kleinmuntz's partial processing, a level of of decision making made with incompletely remembered information and run with approximate values. It is very difficult to accurately simulate how well a player is using the information remembered in this way and is generally represented by a partial-use of related data, as was the case in Sterman and Kleinmuntz's calculations of player memory of supply lines.

In order to accommodate for the time constants, though, players are instead likely using a mental crutch, like that described by Elg during the fire-fighting scenario. After careful inspection of past game data, it can be found that players have a tendency to respond to supply-line changes only after they have 1: exceeded their requirements by wide margin or 2: begun to receive the excess supplies from their suppliers. It seems that either a very large number of over-orders or the presence of the supply line in their turn-to-turn data reminds players of the past excess orders they have made, causing them to stop future ordering. The process is likely unconscious and a simple way to cope with the time-lag present in the face of uncertainty.

What has been described thus-far has been the formulations of the decision making structure behind player response to the supply chain and little more. Still, there are two other features which have a significant impact upon the simulation, neither of which was discussed by Sterman or Kleinmuntz. The first and most significant of these is the sticky-decision-making tendencies of players and people in general during this type of experience. In most long-term situation, people tend to anchor around a decision they have made and delay changes until the requirement has become extreme. In the beer game this can be seen easily in the data, with highly frequent clusters of identical responses, followed by large changes. This trend differs from person-to-person, but makes for a significant additional delay when people play the game, often resulting in an additional turn to respond to problems.

The second is the adjustment of long-term inventory goals over the course of the game. This is a non-traditional risk based behavior where players tend towards a 0-inventory goal as their risk taking tendencies believe the benefit great enough or the situation stable enough to decrease the number. Likewise, in high-risk scenarios the number might increase, or when players in general might view the situation as risky to begin with. Each player's calculation of risk differs, but has several key factors. First of all, as this is a subconscious conceptual process, the information retrieved has a high focus on recent information, likely using a rolling average. If the game went was entirely played with long-term memory this effect might be reduced, but it is traditionally played over a relatively short time period. Secondly, player processing of this perceived risk is based around how easily they are able to maintain desired inventory goals. There is no direct data supporting this theory, however, there is no other structure in the game that the player can realistically use as the basis of risk-assessment. It has already been shown that players have significantly different behavior based upon their risk-taking tendencies.

Risk is calculated by people as a frequency of an event occurring with a weighting based upon how poor the event is. In the case of this game, the 'risk' found would be that of having a backlog, and the penalty would be double that of the size of the backlog present. A reasonable player would likely compare this to the costs of keeping excess inventory, as that is the scoring method of the game, but in practice this is likely not the case. Players generally use original inventory values as their base goals, and would likely then compare risk of failure relative to their current inventory goal. Penalty would be equal to ½ current averaged backlog over the period they are most comfortable with. When the perceived penalty decreased below their threshold of risk, the players would then gradually begin decreasing their functional goal. If players found themselves unable to meet demand in the past, they would likely have a higher inventory goal to better be able to deal with fluctuations in demand. Having the inventory goal increase will cause them to feel they have an increase in control (or at least an attempted increase) and dampen further responses. Having too much will have them dampen their inventory goal or stop changing it. This basic feedback structure will make their penalties self-correcting, stopping them from exploding out of control in either direction.

of sticky-decisions is a frequent one in many social sciences. It has been shown in many ways that people have a tendency to make decisions close to or the same as previous ones unless a clear reason for change has occurred. This can take many forms, but the most relevant one is in the apparent 'grouping' of responses during the beer game, where players tend to change very little about their responses until absolutely necessary. In this case, I have taken this equate to the desired percentage of change^reluctance to change. This means that when playing the game, the player's reluctance to change because of either mental apathy or some other factor is proportional to how minor the change to occur is. In the face of significant changes, such as doubling the number, the change will be fast. Doubling was chosen to be the key tipping-point because it is the basic point in which humans distinguish difference. In tests involving infants, for example, they were able to notice when the number of objects increased to double, but not 1.5 times. The same pattern of mental recognition extends to humans of all ages. It should be noted now, that without a trend recognizing response to changes in demand, this 'sticky response' will take too long to respond to changes. It is also possible to get reasonable responses by increasing the rate in which the 'decision process' is made, however this will increase the strength of the sticky-response overall. If let alone the default delay becomes a full turn for the processing of decision changes, leading to increased delays and inaccuracy over what would normally occur.

The initial goal of this research was to find if one could find a heuristic structure that could replicate the net variability of Sterman's equations, provide reasonable outputs, and better explain poor behavior or at the very least demonstrate different behavior during non-ideal occasions. The reason for this goal was because it was discovered that the Sterman equations were overly robust. They would produce approximately equivalent net variability for a system, even when the customer demand was vastly different or even when key aspects of the system were changed slightly.

The process involved was fairly simple. System Dynamics modeling was first used to recreate the dynamics of the Beer Game itself. The Beer Game, a simple cooperative supply chain game, works well as an example for many of the core concepts behind supply-chain dynamics. After the structure of the game itself was modeled, research was done on the various heuristic structures behind that players might use when playing the beer game. These structures, including the original one created by Sterman, were then modeled and incorporated Game System Dynamics model. The end result was the ability to simulate a large number of possible behaviors for players in the Beer Game. These models were then tested in a large number of situations, and the data was transferred to a spreadsheet where the net variance of each sector during the various conditions was calculated.

The findings of the research do not have far-reaching implications, but they do have interesting ones. The most significant finding, of course, was that the net variance of the Sterman equations are overly robust. Their net variability does not change in any significant way, even with huge differences in initial ordering patterns and significant differences in the very situation itself. It can be argued that they simply reflect the decisions a player had made over the course of a game, but even then their inability to adapt or even produce different behavior leads them to be unrealistic for many situations. Likewise, it was found that the way net variability was affected by conditions was a factor mostly of the heuristic structure being used, and it was found that the conditions of suppliers often was more significant than the ordering sector itself. This leads to the supplying sectors having their net variance change very little, even when customer orders increase dramatically. Partly this is because the variance of their incoming orders is already so great, but it is also in part because they are better able to deal with these huge variations because they have a considerable supply.

Conclusion

The initial goal of this research was to find if one could find a heuristic structure that could replicate the net variability of Sterman's equations, provide reasonable outputs, and better explain poor behavior or at the very least demonstrate different behavior during non-ideal occasions. The reason for this goal was because it was discovered that the Sterman equations were overly robust. They would produce approximately equivalent net variability for a system, even when the customer demand was vastly different or even when key aspects of the system were changed slightly.

The process involved was fairly simple. System Dynamics modeling was first used to recreate the dynamics of the Beer Game itself. The Beer Game, a simple cooperative supply chain game, works well as an example for many of the core concepts behind supply-chain dynamics. After the structure of the game itself was modeled, research was done on the various heuristic structures behind that players might use when playing the beer game. These structures, including the original one created by Sterman, were then modeled and incorporated into the Beer Game System Dynamics model. The end result was the ability to simulate a number of possible behaviors for players in the Beer Game. These models were then tested in a large number of situations, and the data was transferred to a spreadsheet where the net variance of each sector during the various conditions was calculated.

The findings of the research do not have far-reaching implications, but they do have interesting ones. The most significant finding, of course, was that the net variance of the Sterman equations are overly robust. Their net variability does not change in any significant way, even with huge differences in initial ordering patterns and significant differences in the very situation itself. It can be argued that they simply reflect the decisions a player had made over the course of a game, but even then their inability to adapt or even produce different behavior leads them to be unrealistic for many situations. Likewise, it was found that the way net variability was affected by conditions was a factor mostly of the heuristic structure being used, and it was found that the conditions of suppliers often was more significant than the ordering sector itself. This leads to the supplying sectors having their net variance change very little, even when customer orders increase dramatically. Partly this is because the variance of their incoming orders is already so great, but it is also in part because they are better able to deal with these huge variations because they have a constant and dependable supply.

It is recommended that any follow-up work focuses on using these heuristics in order to predict human behavior as conditions change. Such experiments would be tremendously beneficial in determining if these heuristics and indeed indicative of human behavior. Additionally, a more detailed statistical analysis of the model, in order to type-fit data to model parameters would also improve confidence in the model. Either of these would be a tremendous improvement from the current work.

References

Sancar, Urin A. (2003) Quantification of the Bullwhip Effect. Department of Management Information systems, Bogazici University. http://www.mis.boun.edu.tr/erdem/mis517/projects-03/urun.pdf

Chen, F., Drezner, Z., Ryan, J. K., & Simchi-Levi, D. (2000). Quantifying the bullwhip effect in a simple supply chain: The impact of forecasting, lead times, and information. *Management Science, 46*(3), 436-443. Retrieved from http://www.jstor.org/stable/2634741

Don N. Kleinmuntz. (1985). Cognitive heuristics and feedback in a dynamic decision environment. *Management Science, 31*(6), 680-702. Retrieved from http://www.jstor.org/stable/2631444

Kleinmuntz, Don N. (1993). Information processing and misperceptions of the implications of feedback in dynamic decision making. System Dynamics Review Volume 9 Number 3 Fall 1993

Sterman, J. D. (1987). Testing behavioral simulation models by direct experiment. *Management Science, 33*(12), 1572-1592. Retrieved from http://www.jstor.org/stable/2632200

Sterman, J. D. (1989). Modeling managerial behavior: Misperceptions of feedback in a dynamic decision making experiment. *Management Science, 35*(3), 321-339. Retrieved from http://www.jstor.org/stable/2631975

Rigas G., & Elg, F. (1997). *Mental models, confidence, and performance in a complex dynamic decision making environment*. Retrieved on April 1, 2009 from www.ie.boun.edu.tr/labs/sesdyn/isdc97/TURKIA.doc

Brehmer, Berndt. Elg, Fredrik. (2005) *Heuristics in Dynamic Decision Making: Coping with the Time Constants of a Dynamic Task by Doing Something Else* . Retrieved from http://www.systemdynamics.org/conferences/2005/proceed/papers/BREHM340.pdf

May, Synthia P.(1999) Synchrony effects in cognition: The costs and a benefit. Psychonomic Bulletin & Review 1999, 6 (1), 142-147. Retrieved from http://www.psychonomic.org/search/view.cgi?id=2668

Pollitt, E; Leibel, RL; Greenfield, D.(1981) Brief fasting, stress, and cognition in children. American Journal of Clinical Nutrition, Vol 34, 1526-1533 1981. Retrieved from: http://www.ajcn.org/cgi/content/abstract/34/8/1526

Cowan, Nelson.(2000) The magical number 4 in short-term memory: A reconsideration of mental storage capacity, BEHAVIORAL AND BRAIN SCIENCES 24, 87–185. Retrieved from http://journals.cambridge.org/download.php?file=%2FBBS%2FBBS24_01%2FS0140525X01003922a.pdf&code=67a7d81b003b2945b38b329bcd41b762

Tversky, Amos; Kahneman, Daniel. Judgment under Uncertainty: Heuristics and Biases Science, New Series, Vol 185 No 4157 Sep 27, 1974 pp1124-1131 Retrieved from http://links.jstor.org/sici?sici=0036-8075%2819740927%293%3A185%3A4157%3C1124%3AJUUHAB%3E2.0.CO%3B2-M

Otto, Andreas. Herbert Kotzab; Does supply chain management really pay? Six perspectives to measure the performance of managing a supply chain European Journal of Operational Research Volume 144, Issue 2, 16 January 2003, Pages 306-320 www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6VCT-46Y59H7-1&_user=10&_rdoc=1&_fmt=&_orig=search&_sort=d&_docanchor=&view=c&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=76e6d2810d82e4de79a1203128f0a4ca

Bussiness Dynamics: systems thinking and modeling for a complex world / John D. Sterman Irwin/McGraw-Hill, 2000

Appendix 1: **Source of Illustrations Used**

Illustration 32:Unified Model - Isolated Behavior
Source: Kevin Piala (author) BeerGame Simple 2

Illustration 33:Unified Model Structure
Source: Kevin Piala (author) BeerGame Simple 2

Illustration 34:Unified Model- Complete Behavior
Source: Kevin Piala (author) BeerGame Simple 2

Appendix 2: ***Equations for the BeerGame Simplified 2 Model***

(001) Anchor Backlog= INTEG (
Anchor Backlog Change,
0)
Units: Units

(002) Anchor Backlog 0= INTEG (
Anchor Backlog Change 1,
0)
Units: Units

(003) Anchor Backlog 1= INTEG (
Anchor Backlog Change 2,
0)
Units: Units

(004) Anchor Backlog 2= INTEG (
Anchor Backlog Change 3,
0)
Units: Units

(005) Anchor Backlog 3= INTEG (
Anchor Backlog Change 4,
0)
Units: Units

(006) Anchor Backlog Change=
Anchor Incoming Orders-Outgoing Anchor
Units: Units/Turn

(007) Anchor Backlog Change 0=
Anchor Complex Incoming Orders-Outgoing Anchor Complex
Units: Units/Turn

(008) Anchor Backlog Change 0 0=
Anchor Complex Incoming Orders 0-Outgoing Anchor Complex 0
Units: Units/Turn

(009) Anchor Backlog Change 0 1=
Anchor Complex Incoming Orders 1-Outgoing Anchor Complex 1
Units: Units/Turn

(010) Anchor Backlog Change 0 2=
Anchor Complex Incoming Orders 2-Outgoing Anchor Complex 2
Units: Units/Turn

(011) Anchor Backlog Change 0 3=

Anchor Complex Incoming Orders 3-Outgoing Anchor Complex 3
Units: Units/Turn

(012) Anchor Backlog Change 1=
Anchor Incoming Orders 0-Outgoing Anchor 0
Units: Units/Turn

(013) Anchor Backlog Change 2=
Anchor Incoming Orders 1-Outgoing Anchor 1
Units: Units/Turn

(014) Anchor Backlog Change 3=
Anchor Incoming Orders 2-Outgoing Anchor 2
Units: Units/Turn

(015) Anchor Backlog Change 4=
Anchor Incoming Orders 3-Outgoing Anchor 3
Units: Units/Turn

(016) Anchor Complex Backlog= INTEG (
Anchor Backlog Change 0,
0)
Units: Units

(017) Anchor Complex Backlog 0= INTEG (
Anchor Backlog Change 0 0,
0)
Units: Units

(018) Anchor Complex Backlog 1= INTEG (
Anchor Backlog Change 0 1,
0)
Units: Units

(019) Anchor Complex Backlog 2= INTEG (
Anchor Backlog Change 0 2,
0)
Units: Units

(020) Anchor Complex Backlog 3= INTEG (
Anchor Backlog Change 0 3,
0)
Units: Units

(021) Anchor Complex Caution=
*Anchor Complex Supply Line*Anchor Complex Supply Line Accounting*
Units: Units/Turn/Turn

(022) Anchor Complex Caution 0=
*Anchor Complex Supply Line 0*Anchor Complex Supply Line Accounting 0*
Units: Units/Turn/Turn

(023) Anchor Complex Caution 1=
*Anchor Complex Supply Line 1*Anchor Complex Supply Line Accounting 1*
Units: Units/Turn/Turn

(024) Anchor Complex Caution 2=
*Anchor Complex Supply Line 2*Anchor Complex Supply Line Accounting 2*
Units: Units/Turn/Turn

(025) Anchor Complex Caution 3=
*Anchor Complex Supply Line 3*Anchor Complex Supply Line Accounting 3*
Units: Units/Turn/Turn

(026) Anchor Complex Desired Inventory=
12
Units: Units

(027) Anchor Complex Desired Inventory 0=
12
Units: Units

(028) Anchor Complex Desired Inventory 1=
12
Units: Units

(029) Anchor Complex Desired Inventory 2=
12
Units: Units

(030) Anchor Complex Desired Inventory 3=
12
Units: Units

(031) Anchor Complex Desired Supply Line 0=
*Anchor Complex Incoming Orders*Rate of Return 1 0*
Units: Units

(032) Anchor Complex Desired Supply Line 0 0=
*Anchor Complex Incoming Orders 0*Rate of Return 1 0 0*
Units: Units

(033) Anchor Complex Desired Supply Line 0 1=
*Anchor Complex Incoming Orders 1*Rate of Return 1 0 1*

Units: Units

(034) *Anchor Complex Desired Supply Line 0 2=*
*Anchor Complex Incoming Orders 2*Rate of Return 1 0 2*
Units: Units

(035) *Anchor Complex Desired Supply Line 0 3=*
*Anchor Complex Incoming Orders 3*Rate of Return 1 0 3*
Units: Units

(036) *Anchor Complex Desired Ordering=*
MAX(0, (Anchor Complex Incoming Orders-(Anchor Complex Inventory Adjust Rate
**Anchor Complex Inventory Discrepancy)))*
Units: Units/Turn

(037) *Anchor Complex Desired Ordering 0=*
MAX(0, (Anchor Complex Incoming Orders 0-(Anchor Complex Inventory Adjust Rate 0
**Anchor Complex Inventory Discrepancy 0)))*
Units: Units/Turn

(038) *Anchor Complex Desired Ordering 1=*
MAX(0, (Anchor Complex Incoming Orders 1-(Anchor Complex Inventory Adjust Rate 1
**Anchor Complex Inventory Discrepancy 1)))*
Units: Units/Turn

(039) *Anchor Complex Desired Ordering 2=*
MAX(0, (Anchor Complex Incoming Orders 2-(Anchor Complex Inventory Adjust Rate 2
**Anchor Complex Inventory Discrepancy 2)))*
Units: Units/Turn

(040) *Anchor Complex Desired Ordering 3=*
MAX(0, (Anchor Complex Incoming Orders 3-(Anchor Complex Inventory Adjust Rate 3
**Anchor Complex Inventory Discrepancy 3)))*
Units: Units/Turn

(041) *Anchor Complex effective Inventory=*
Anchor Complex Inventory-Anchor Complex Backlog
Units: Units

(042) *Anchor Complex effective Inventory 0=*
Anchor Complex Inventory 0-Anchor Complex Backlog 0
Units: Units

(043) *Anchor Complex effective Inventory 1=*
Anchor Complex Inventory 1-Anchor Complex Backlog 1
Units: Units

(044) Anchor Complex effective Inventory 2=
Anchor Complex Inventory 2-Anchor Complex Backlog 2
Units: Units

(045) Anchor Complex effective Inventory 3=
Anchor Complex Inventory 3-Anchor Complex Backlog 3
Units: Units

(046) Anchor Complex Incoming Orders=
4+STEP(4, 4)
Units: Units/Turn

(047) Anchor Complex Incoming Orders 0=
*(4+STEP(4, 4))*2*
Units: Units/Turn

(048) Anchor Complex Incoming Orders 1=
DELAY FIXED(Anchor Complex Ordering 0, 2, 4)
Units: Units/Turn

(049) Anchor Complex Incoming Orders 2=
DELAY FIXED(Anchor Complex Ordering 1, 2, 4)
Units: Units/Turn

(050) Anchor Complex Incoming Orders 3=
DELAY FIXED(Anchor Complex Ordering 2, 2, 4)
Units: Units/Turn

(051) Anchor Complex Incoming Shipments=
Anchor Complex Ordering
Units: Units/Turn

(052) Anchor Complex Incoming Shipments 0=
Outgoing Anchor Complex 1
Units: Units/Turn

(053) Anchor Complex Incoming Shipments 1=
Outgoing Anchor Complex 2
Units: Units/Turn

(054) Anchor Complex Incoming Shipments 2=
Outgoing Anchor Complex 3
Units: Units/Turn

(055) Anchor Complex Incoming Shipments 3=
Anchor Complex Ordering 3
Units: Units/Turn

```
(056)   Anchor Complex Inventory= INTEG (
                Incoming Anchor Complex-Outgoing Anchor Complex,
                        12)
        Units: Units

(057)   Anchor Complex Inventory 0= INTEG (
                Incoming Anchor Complex 0-Outgoing Anchor Complex 0,
                        12)
        Units: Units

(058)   Anchor Complex Inventory 1= INTEG (
                Incoming Anchor Complex 1-Outgoing Anchor Complex 1,
                        12)
        Units: Units

(059)   Anchor Complex Inventory 2= INTEG (
                Incoming Anchor Complex 2-Outgoing Anchor Complex 2,
                        12)
        Units: Units

(060)   Anchor Complex Inventory 3= INTEG (
                Incoming Anchor Complex 3-Outgoing Anchor Complex 3,
                        12)
        Units: Units

(061)   Anchor Complex Inventory Adjust Rate=
                0.5
        Units: 1/Turn

(062)   Anchor Complex Inventory Adjust Rate 0=
                0.1
        Units: 1/Turn

(063)   Anchor Complex Inventory Adjust Rate 1=
                0.1
        Units: 1/Turn

(064)   Anchor Complex Inventory Adjust Rate 2=
                0.1
        Units: 1/Turn

(065)   Anchor Complex Inventory Adjust Rate 3=
                0.1
        Units: 1/Turn

(066)   Anchor Complex Inventory Discrepancy=
```

Anchor Complex effective Inventory-Anchor Complex Desired Inventory
Units: Units

(067) Anchor Complex Inventory Discrepancy 0=
Anchor Complex effective Inventory 0-Anchor Complex Desired Inventory 0
Units: Units

(068) Anchor Complex Inventory Discrepancy 1=
Anchor Complex effective Inventory 1-Anchor Complex Desired Inventory 1
Units: Units

(069) Anchor Complex Inventory Discrepancy 2=
Anchor Complex effective Inventory 2-Anchor Complex Desired Inventory 2
Units: Units

(070) Anchor Complex Inventory Discrepancy 3=
Anchor Complex effective Inventory 3-Anchor Complex Desired Inventory 3
Units: Units

(071) Anchor Complex Memory Loss=
Anchor Complex Supply Line/Memory Loss Time
Units: Units/Turn

(072) Anchor Complex Memory Loss 0=
Anchor Complex Supply Line 0/Memory Loss Time 0
Units: Units/Turn

(073) Anchor Complex Memory Loss 1=
Anchor Complex Supply Line 1/Memory Loss Time 1
Units: Units/Turn

(074) Anchor Complex Memory Loss 2=
Anchor Complex Supply Line 2/Memory Loss Time 2
Units: Units/Turn

(075) Anchor Complex Memory Loss 3=
Anchor Complex Supply Line 3/Memory Loss Time 3
Units: Units/Turn

(076) Anchor Complex Ordering=
MAX(0, Anchor Complex Ordering a)
Units: Units/Turn

(077) Anchor Complex Ordering 0=
MAX(0, Anchor Complex Ordering a 0)
Units: Units/Turn

(078) Anchor Complex Ordering 1=
MAX(0, Anchor Complex Ordering a 1)
Units: Units/Turn

(079) Anchor Complex Ordering 2=
MAX(0, Anchor Complex Ordering a 2)
Units: Units/Turn

(080) Anchor Complex Ordering 3=
MAX(0, Anchor Complex Ordering a 3)
Units: Units/Turn

(081) Anchor Complex Ordering a= INTEG (
Change in Anchor Complex Ordering-Anchor Complex Caution,
4)
Units: Units/Turn

(082) Anchor Complex Ordering a 0= INTEG (
Change in Anchor Complex Ordering 0-Anchor Complex Caution 0,
4)
Units: Units/Turn

(083) Anchor Complex Ordering a 1= INTEG (
Change in Anchor Complex Ordering 1-Anchor Complex Caution 1,
4)
Units: Units/Turn

(084) Anchor Complex Ordering a 2= INTEG (
Change in Anchor Complex Ordering 2-Anchor Complex Caution 2,
4)
Units: Units/Turn

(085) Anchor Complex Ordering a 3= INTEG (
Change in Anchor Complex Ordering 3-Anchor Complex Caution 3,
4)
Units: Units/Turn

(086) Anchor Complex Supply Line= INTEG (
Change in Anchor Complex Supply Line-Anchor Complex Memory Loss,
16)
Units: Units

(087) Anchor Complex Supply Line 0= INTEG (
Change in Anchor Complex Supply Line 0-Anchor Complex Memory Loss 0,
0)
Units: Units

(088) Anchor Complex Supply Line 1= INTEG (
Change in Anchor Complex Supply Line 1-Anchor Complex Memory Loss 1,
0)
Units: Units

(089) Anchor Complex Supply Line 2= INTEG (
Change in Anchor Complex Supply Line 2-Anchor Complex Memory Loss 2,
0)
Units: Units

(090) Anchor Complex Supply Line 3= INTEG (
Change in Anchor Complex Supply Line 3-Anchor Complex Memory Loss 3,
0)
Units: Units

(091) Anchor Complex Supply Line Accounting=
1
Units: 1/Turn/Turn

(092) Anchor Complex Supply Line Accounting 0=
1
Units: 1/Turn/Turn

(093) Anchor Complex Supply Line Accounting 1=
1
Units: 1/Turn/Turn

(094) Anchor Complex Supply Line Accounting 2=
1
Units: 1/Turn/Turn

(095) Anchor Complex Supply Line Accounting 3=
1
Units: 1/Turn/Turn

(096) Anchor Desired Inventory=
12
Units: Units

(097) Anchor Desired Inventory 0=
12
Units: Units

(098) Anchor Desired Inventory 1=
12
Units: Units

(099) Anchor Desired Inventory 2=
12
Units: Units

(100) Anchor Desired Inventory 3=
12
Units: Units

(101) Anchor Desired Supply Line=
*Anchor Incoming Orders*Rate of Return 1*
Units: Units

(102) Anchor Desired Ordering=
*MAX(0, -Anchor Inventory Discrepancy*Anchor Inventory Adjust Rate+Anchor Incoming Orders*
)
Units: Units/Turn

(103) Anchor Desired Ordering 0=
*MAX(0, -Anchor Inventory Discrepancy 0*Anchor Inventory Adjust Rate 0+Anchor Incoming Orders 0*
)
Units: Units/Turn

(104) Anchor Desired Ordering 1=
*MAX(0, -Anchor Inventory Discrepancy 1*Anchor Inventory Adjust Rate 1+Anchor Incoming Orders 1*
)
Units: Units/Turn

(105) Anchor Desired Ordering 2=
*MAX(0, -Anchor Inventory Discrepancy 2*Anchor Inventory Adjust Rate 2+Anchor Incoming Orders 2*
)
Units: Units/Turn

(106) Anchor Desired Ordering 3=
*MAX(0, -Anchor Inventory Discrepancy 3*Anchor Inventory Adjust Rate 3+Anchor Incoming Orders 3*
)
Units: Units/Turn

(107) Anchor effective Inventory=
Anchor Inventory-Anchor Backlog
Units: Units

(108) Anchor effective Inventory 0=

Anchor Inventory 0-Anchor Backlog 0
Units: Units

(109) Anchor effective Inventory 1=
Anchor Inventory 1-Anchor Backlog 1
Units: Units

(110) Anchor effective Inventory 2=
Anchor Inventory 2-Anchor Backlog 2
Units: Units

(111) Anchor effective Inventory 3=
Anchor Inventory 3-Anchor Backlog 3
Units: Units

(112) Anchor Incoming Orders=
4+STEP(4, 4)
Units: Units/Turn

(113) Anchor Incoming Orders 0=
4+STEP(4, 4)
Units: Units/Turn

(114) Anchor Incoming Orders 1=
DELAY FIXED(Anchor Ordering 0, 2, 4)
Units: Units/Turn

(115) Anchor Incoming Orders 2=
DELAY FIXED(Anchor Ordering 1, 2, 4)
Units: Units/Turn

(116) Anchor Incoming Orders 3=
DELAY FIXED(Anchor Ordering 2, 2, 4)
Units: Units/Turn

(117) Anchor Incoming Shipments=
Anchor Ordering
Units: Units/Turn

(118) Anchor Incoming Shipments 0=
Outgoing Anchor 1
Units: Units/Turn

(119) Anchor Incoming Shipments 1=
Outgoing Anchor 2
Units: Units/Turn

(120) Anchor Incoming Shipments 2=
Outgoing Anchor 3
Units: Units/Turn

(121) Anchor Incoming Shipments 3=
Anchor Ordering 3
Units: Units/Turn

(122) Anchor Inventory= INTEG (
Incoming Anchor-Outgoing Anchor,
12)
Units: Units

(123) Anchor Inventory 0= INTEG (
Incoming Anchor 0-Outgoing Anchor 0,
12)
Units: Units

(124) Anchor Inventory 1= INTEG (
Incoming Anchor 1-Outgoing Anchor 1,
12)
Units: Units

(125) Anchor Inventory 2= INTEG (
Incoming Anchor 2-Outgoing Anchor 2,
12)
Units: Units

(126) Anchor Inventory 3= INTEG (
Incoming Anchor 3-Outgoing Anchor 3,
12)
Units: Units

(127) Anchor Inventory Adjust Rate=
1
Units: 1/Turn

(128) Anchor Inventory Adjust Rate 0=
1
Units: 1/Turn

(129) Anchor Inventory Adjust Rate 1=
1
Units: 1/Turn

(130) Anchor Inventory Adjust Rate 2=
1

```
        Units: 1/Turn

(131)   Anchor Inventory Adjust Rate 3=
                1
        Units: 1/Turn

(132)   Anchor Inventory Discrepancy=
                Anchor effective Inventory-Anchor Desired Inventory
        Units: Units

(133)   Anchor Inventory Discrepancy 0=
                Anchor effective Inventory 0-Anchor Desired Inventory 0
        Units: Units

(134)   Anchor Inventory Discrepancy 1=
                Anchor effective Inventory 1-Anchor Desired Inventory 1
        Units: Units

(135)   Anchor Inventory Discrepancy 2=
                Anchor effective Inventory 2-Anchor Desired Inventory 2
        Units: Units

(136)   Anchor Inventory Discrepancy 3=
                Anchor effective Inventory 3-Anchor Desired Inventory 3
        Units: Units

(137)   Anchor Ordering= INTEG (
                Change in Anchor Ordering,
                        4)
        Units: Units/Turn

(138)   Anchor Ordering 0= INTEG (
                Change in Anchor Ordering 0,
                        4)
        Units: Units/Turn

(139)   Anchor Ordering 1= INTEG (
                Change in Anchor Ordering 1,
                        4)
        Units: Units/Turn

(140)   Anchor Ordering 2= INTEG (
                Change in Anchor Ordering 2,
                        4)
        Units: Units/Turn

(141)   Anchor Ordering 3= INTEG (
```

Change in Anchor Ordering 3,
4)
Units: Units/Turn

(142) Anchor Supply Line= INTEG (
Change in Anchor Supply Line,
16)
Units: Units

(143) Backlog Change=
Customer Orders-Shipments from Retailer
Units: Units/Turn

(144) Backlog Change 0=
Customer Orders 0-Shipments from Retailer 0
Units: Units/Turn

(145) Backlog Change 1=
Customer Orders 1-Shipments from Retailer 1
Units: Units/Turn

(146) Backlog Change 2=
Customer Orders 2-Shipments from Retailer 2
Units: Units/Turn

(147) Backlog Change 3=
Customer Orders 3-Shipments from Retailer 3
Units: Units/Turn

(148) Backlog Retailer= INTEG (
Backlog Change,
0)
Units: Units

(149) Backlog Retailer 0= INTEG (
Backlog Change 0,
0)
Units: Units

(150) Backlog Retailer 1= INTEG (
Backlog Change 1,
0)
Units: Units

(151) Backlog Retailer 2= INTEG (
Backlog Change 2,
0)

Units: Units

(152) Backlog Retailer 3= INTEG (
Backlog Change 3,
0)
Units: Units

(153) Change in Anchor Complex Ordering=
(Anchor Complex Desired Ordering-Anchor Complex Ordering a)/Time to Change
Units: Units/Turn/Turn

(154) Change in Anchor Complex Ordering 0=
(Anchor Complex Desired Ordering 0-Anchor Complex Ordering a 0)/Time to Change 0
Units: Units/Turn/Turn

(155) Change in Anchor Complex Ordering 1=
(Anchor Complex Desired Ordering 1-Anchor Complex Ordering a 1)/Time to Change 1
Units: Units/Turn/Turn

(156) Change in Anchor Complex Ordering 2=
(Anchor Complex Desired Ordering 2-Anchor Complex Ordering a 2)/Time to Change 2
Units: Units/Turn/Turn

(157) Change in Anchor Complex Ordering 3=
(Anchor Complex Desired Ordering 3-Anchor Complex Ordering a 3)/Time to Change 3
Units: Units/Turn/Turn

(158) Change in Anchor Complex Supply Line=
Anchor Complex Ordering-Anchor Complex Incoming Orders
Units: Units/Turn

(159) Change in Anchor Complex Supply Line 0=
Anchor Complex Ordering 0-Anchor Complex Incoming Orders 0
Units: Units/Turn

(160) Change in Anchor Complex Supply Line 1=
Anchor Complex Ordering 1-Anchor Complex Incoming Orders 1
Units: Units/Turn

(161) Change in Anchor Complex Supply Line 2=
Anchor Complex Ordering 2-Anchor Complex Incoming Orders 2
Units: Units/Turn

(162) Change in Anchor Complex Supply Line 3=
Anchor Complex Ordering 3-Anchor Complex Incoming Orders 3
Units: Units/Turn

(163) Change in Anchor Ordering=
(Anchor Desired Ordering-Anchor Ordering)/Order change Rate
Units: Units/Turn/Turn

(164) Change in Anchor Ordering 0=
(Anchor Desired Ordering 0-Anchor Ordering 0)/Order change Rate 0
Units: Units/Turn/Turn

(165) Change in Anchor Ordering 1=
(Anchor Desired Ordering 1-Anchor Ordering 1)/Order change Rate 1
Units: Units/Turn/Turn

(166) Change in Anchor Ordering 2=
(Anchor Desired Ordering 2-Anchor Ordering 2)/Order change Rate 2
Units: Units/Turn/Turn

(167) Change in Anchor Ordering 3=
(Anchor Desired Ordering 3-Anchor Ordering 3)/Order change Rate 3
Units: Units/Turn/Turn

(168) Change in Anchor Supply Line=
Anchor Ordering-Incoming Anchor
Units: Units/Turn

(169) Change in Kleinmuntz Supply=
Kleinmuntz Ordering-Incoming Klien
Units: Units/Turn

(170) Change in Kleinmuntz Supply 0=
Kleinmuntz Ordering 0-Incoming Klien 0
Units: Units/Turn

(171) Change in Kleinmuntz Supply 1=
Kleinmuntz Ordering 1-Incoming Klien 1
Units: Units/Turn

(172) Change in Kleinmuntz Supply 2=
Kleinmuntz Ordering 2-Incoming Klien 2
Units: Units/Turn

(173) Change in Kleinmuntz Supply 3=
Kleinmuntz Ordering 3-Incoming Klien 3
Units: Units/Turn

(174) Change in Perceived change in inventory=
(Change in Perceived Inventory-Perceived Change in Inventory)/Time to change view
Units: Units/Turn/Turn

(175) Change in Perceived change in inventory 0=
(Change in Perceived Inventory 0-Perceived Change in Inventory 0)/Time to change view 0
Units: Units/Turn/Turn

(176) Change in Perceived change in inventory 1=
(Change in Perceived Inventory 1-Perceived Change in Inventory 1)/Time to change view 1
Units: Units/Turn/Turn

(177) Change in Perceived change in inventory 2=
(Change in Perceived Inventory 2-Perceived Change in Inventory 2)/Time to change view 2
Units: Units/Turn/Turn

(178) Change in Perceived change in inventory 3=
(Change in Perceived Inventory 3-Perceived Change in Inventory 3)/Time to change view 3
Units: Units/Turn/Turn

(179) Change in Perceived Inventory=
(Safe effective Inventory-Perceived Effective Inventory)/Time to change view
Units: Units/Turn

(180) Change in Perceived Inventory 0=
(Safe effective Inventory 0-Perceived Effective Inventory 0)/Time to change view 0
Units: Units/Turn

(181) Change in Perceived Inventory 1=
(Safe effective Inventory 1-Perceived Effective Inventory 1)/Time to change view 1
Units: Units/Turn

(182) Change in Perceived Inventory 2=
(Safe effective Inventory 2-Perceived Effective Inventory 2)/Time to change view 2
Units: Units/Turn

(183) Change in Perceived Inventory 3=
(Safe effective Inventory 3-Perceived Effective Inventory 3)/Time to change view 3
Units: Units/Turn

(184) Change in Supply Line=
Sterman Ordering-Incoming Sterman
Units: Units/Turn

(185) Change in Supply Line 0=
Sterman Ordering 0-Incoming Sterman 0

Units: Units/Turn

(186) Change in Supply Line 1=
Sterman Ordering 1-Incoming Sterman 1
Units: Units/Turn

(187) Change in Supply Line 2=
Sterman Ordering 2-Incoming Sterman 2
Units: Units/Turn

(188) Change in Supply Line 3=
Sterman Ordering 3-Incoming Sterman 3
Units: Units/Turn

(189) Customer Orders=
Initial Orders(1+STEP(1, 5))+STEP(0, 5)*
Units: Units/Turn

(190) Customer Orders 0=
DELAY FIXED(Retailer Orders, 2 , Initial Orders 0)
Units: Units/Turn

(191) Customer Orders 1=
DELAY FIXED(Retailer Orders 0, 2 , Initial Orders 1)
Units: Units/Turn

(192) Customer Orders 2=
DELAY FIXED(Retailer Orders 1, 2 , Initial Orders 2)
Units: Units/Turn

(193) Customer Orders 3=
Initial Orders 3(1+STEP(1, 5))+STEP(0, 5)*
Units: Units/Turn

(194) Desired Inventory=
12
Units: Units

(195) Desired Inventory 0=
12
Units: Units

(196) Desired Inventory 1=
12
Units: Units

(197) Desired Inventory 2=

12
Units: Units

(198) Desired Inventory 3=
12
Units: Units

(199) FINAL TIME = 100
Units: Turns
The final time for the simulation.

(200) Incoming Anchor=
DELAY FIXED(Anchor Incoming Shipments, 4, 4)
Units: Units/Turn

(201) Incoming Anchor 0=
DELAY FIXED(Anchor Incoming Shipments 0, 4, 4)
Units: Units/Turn

(202) Incoming Anchor 1=
DELAY FIXED(Anchor Incoming Shipments 1, 4, 4)
Units: Units/Turn

(203) Incoming Anchor 2=
DELAY FIXED(Anchor Incoming Shipments 2, 4, 4)
Units: Units/Turn

(204) Incoming Anchor 3=
DELAY FIXED(Anchor Incoming Shipments 3, 2, 4)
Units: Units/Turn

(205) Incoming Anchor Complex=
DELAY FIXED(Anchor Complex Incoming Shipments, 4, 4)
Units: Units/Turn

(206) Incoming Anchor Complex 0=
DELAY FIXED(Anchor Complex Incoming Shipments 0, 2, 4)
Units: Units/Turn

(207) Incoming Anchor Complex 1=
DELAY FIXED(Anchor Complex Incoming Shipments 1, 2, 4)
Units: Units/Turn

(208) Incoming Anchor Complex 2=
DELAY FIXED(Anchor Complex Incoming Shipments 2, 2, 4)
Units: Units/Turn

(209) Incoming Anchor Complex 3=
DELAY FIXED(Anchor Complex Incoming Shipments 3, 2, 4)
Units: Units/Turn

(210) Incoming Klien=
DELAY FIXED(Klein Incoming Shipments, 4, 10)
Units: Units/Turn

(211) Incoming Klien 0=
DELAY FIXED(Klein Incoming Shipments 0, 4, 4)
Units: Units/Turn

(212) Incoming Klien 1=
DELAY FIXED(Klein Incoming Shipments 1, 4, 4)
Units: Units/Turn

(213) Incoming Klien 2=
DELAY FIXED(Klein Incoming Shipments 2, 4, 4)
Units: Units/Turn

(214) Incoming Klien 3=
DELAY FIXED(Klein Incoming Shipments 3, 2, 4)
Units: Units/Turn

(215) Incoming Safe=
DELAY FIXED(Safe Incoming Shipments, 4, 4)
Units: Units/Turn

(216) Incoming Safe 0=
DELAY FIXED(Safe Incoming Shipments 0, 2, 4)
Units: Units/Turn

(217) Incoming Safe 1=
DELAY FIXED(Safe Incoming Shipments 1, 4, 4)
Units: Units/Turn

(218) Incoming Safe 2=
DELAY FIXED(Safe Incoming Shipments 2, 2, 4)
Units: Units/Turn

(219) Incoming Safe 3=
DELAY FIXED(Safe Incoming Shipments 3, 2, 4)
Units: Units/Turn

(220) Incoming Sterman=
DELAY FIXED(Sterman Incoming Shipments, 4, 4)
Units: Units/Turn

(221) Incoming Sterman 0=
DELAY FIXED(Sterman Incoming Shipments 0, 2, 4)
Units: Units/Turn

(222) Incoming Sterman 1=
DELAY FIXED(Sterman Incoming Shipments 1, 2, 4)
Units: Units/Turn

(223) Incoming Sterman 2=
DELAY FIXED(Sterman Incoming Shipments 2, 2, 4)
Units: Units/Turn

(224) Incoming Sterman 3=
DELAY FIXED(Sterman Incoming Shipments 3, 2, 4)
Units: Units/Turn

(225) Initial Orders=
4
Units: Units/Turn

(226) Initial Orders 0=
4
Units: Units/Turn

(227) Initial Orders 1=
4
Units: Units/Turn

(228) Initial Orders 2=
4
Units: Units/Turn

(229) Initial Orders 3=
4
Units: Units/Turn

(230) INITIAL TIME = 0
Units: Turns
The initial time for the simulation.

(231) Inventory Adjustment Factor=
0.5
Units: 1/Turns

(232) Inventory Adjustment Factor 0=
0.5

Units: 1/Turns

(233) Inventory Adjustment Factor 1=
0.5
Units: 1/Turns

(234) Inventory Adjustment Factor 2=
0.5
Units: 1/Turns

(235) Inventory Adjustment Factor 3=
0.5
Units: 1/Turns

(236) Inventory Correction Rate=
0.25
Units: 1/Turn

(237) Inventory Correction Rate 0=
0.25
Units: 1/Turn

(238) Inventory Correction Rate 1=
0.25
Units: 1/Turn

(239) Inventory Correction Rate 2=
0.25
Units: 1/Turn

(240) Inventory Correction Rate 3=
0.25
Units: 1/Turn

(241) Klein Backlog= INTEG (
Klein Backlog Change,
0)
Units: Units

(242) Klein Backlog 0= INTEG (
Klein Backlog Change 0,
0)
Units: Units

(243) Klein Backlog 1= INTEG (
Klein Backlog Change 1,
0)

```
        Units: Units

(244)   Klein Backlog 2= INTEG (
                Klein Backlog Change 2,
                        0)
        Units: Units

(245)   Klein Backlog 3= INTEG (
                Klein Backlog Change 3,
                        0)
        Units: Units

(246)   Klein Backlog Change=
                Klein Incoming Orders-Outgoing Klien
        Units: Units/Turn

(247)   Klein Backlog Change 0=
                Klein Incoming Orders 0-Outgoing Klien 0
        Units: Units/Turn

(248)   Klein Backlog Change 1=
                Klein Incoming Orders 1-Outgoing Klien 1
        Units: Units/Turn

(249)   Klein Backlog Change 2=
                Klein Incoming Orders 2-Outgoing Klien 2
        Units: Units/Turn

(250)   Klein Backlog Change 3=
                Klein Incoming Orders 3-Outgoing Klien 3
        Units: Units/Turn

(251)   Klein Desired Inventory=
                12
        Units: Units

(252)   Klein Desired Inventory 0=
                12
        Units: Units

(253)   Klein Desired Inventory 1=
                12
        Units: Units

(254)   Klein Desired Inventory 2=
                12
        Units: Units
```

(255) Klein Desired Inventory 3=
12
Units: Units

(256) Klein Desired Supply Line=
*Klein Incoming Orders*Rate of Return 0*
Units: Units

(257) Klein Desired Supply Line 0=
*Klein Incoming Orders 0*Rate of Return 0 0*
Units: Units

(258) Klein Desired Supply Line 1=
*Klein Incoming Orders 1*Rate of Return 0 1*
Units: Units

(259) Klein Desired Supply Line 2=
*Klein Incoming Orders 2*Rate of Return 0 2*
Units: Units

(260) Klein Desired Supply Line 3=
*Klein Incoming Orders 3*Rate of Return 0 3*
Units: Units

(261) Klein Incoming Orders=
*(5+STEP(5, 4))*2*
Units: Units/Turn

(262) Klein Incoming Orders 0=
4+STEP(4, 4)
Units: Units/Turn

(263) Klein Incoming Orders 1=
DELAY FIXED(Kleinmuntz Ordering 0, 2, 4)
Units: Units/Turn

(264) Klein Incoming Orders 2=
DELAY FIXED(Kleinmuntz Ordering 1, 2, 4)
Units: Units/Turn

(265) Klein Incoming Orders 3=
DELAY FIXED(Kleinmuntz Ordering 2, 2, 4)
Units: Units/Turn

(266) Klein Incoming Shipments=
Kleinmuntz Ordering

Units: Units/Turn

(267) *Klein Incoming Shipments 0=*
Outgoing Klien 1
Units: Units/Turn

(268) *Klein Incoming Shipments 1=*
Outgoing Klien 2
Units: Units/Turn

(269) *Klein Incoming Shipments 2=*
Outgoing Klien 3
Units: Units/Turn

(270) *Klein Incoming Shipments 3=*
Kleinmuntz Ordering 3
Units: Units/Turn

(271) *Klein Inventory Adjust Rate=*
0.2
Units: 1/Turn

(272) *Klein Inventory Adjust Rate 0=*
0.1
Units: 1/Turn

(273) *Klein Inventory Adjust Rate 1=*
0.1
Units: 1/Turn

(274) *Klein Inventory Adjust Rate 2=*
0.1
Units: 1/Turn

(275) *Klein Inventory Adjust Rate 3=*
0.1
Units: 1/Turn

(276) *Klein Inventory Discrepancy=*
Klien effective Inventory-Klein Desired Inventory
Units: Units

(277) *Klein Inventory Discrepancy 0=*
Klien effective Inventory 0-Klein Desired Inventory 0
Units: Units

(278) *Klein Inventory Discrepancy 1=*

```
            Klien effective Inventory 1-Klein Desired Inventory 1
        Units: Units

(279)   Klein Inventory Discrepancy 2=
            Klien effective Inventory 2-Klein Desired Inventory 2
        Units: Units

(280)   Klein Inventory Discrepancy 3=
            Klien effective Inventory 3-Klein Desired Inventory 3
        Units: Units

(281)   "Klein Supply-Line Accounting 0"=
            0.5
        Units: 1

(282)   "Klein Supply-Line Accounting 1"=
            0.5
        Units: 1

(283)   "Klein Supply-Line Accounting 2"=
            0.5
        Units: 1

(284)   "Klein Supply-Line Accounting 3"=
            0.5
        Units: 1

(285)   "Klein Supply-Line Accounting"=
            2
        Units: 1

(286)   Kleinmuntz Inventory= INTEG (
            Incoming Klien-Outgoing Klien,
                12)
        Units: Units

(287)   Kleinmuntz Inventory 0= INTEG (
            Incoming Klien 0-Outgoing Klien 0,
                12)
        Units: Units

(288)   Kleinmuntz Inventory 1= INTEG (
            Incoming Klien 1-Outgoing Klien 1,
                12)
        Units: Units

(289)   Kleinmuntz Inventory 2= INTEG (
```

Incoming Klien 2-Outgoing Klien 2,
12)
Units: Units

(290) Kleinmuntz Inventory 3= INTEG (
Incoming Klien 3-Outgoing Klien 3,
12)
Units: Units

(291) Kleinmuntz Ordering=
MAX(0, Klein Incoming Orders+Klein Inventory Adjust Rate("Klein Supply-Line Accounting"*
**(Klein Desired Supply Line-Kleinmuntz Perceived Supplyline)-Klein Inventory Discrepancy*
))
Units: Units/Turn

(292) Kleinmuntz Ordering 0=
MAX(0, Klein Incoming Orders 0+Klein Inventory Adjust Rate 0("Klein Supply-Line Accounting 0"*
**(Klein Desired Supply Line 0-Kleinmuntz Perceived Supplyline 0)-Klein Inventory Discrepancy 0*
))
Units: Units/Turn

(293) Kleinmuntz Ordering 1=
MAX(0, Klein Incoming Orders 1+Klein Inventory Adjust Rate 1("Klein Supply-Line Accounting 1"*
**(Klein Desired Supply Line 1-Kleinmuntz Perceived Supplyline 1)-Klein Inventory Discrepancy 1*
))
Units: Units/Turn

(294) Kleinmuntz Ordering 2=
MAX(0, Klein Incoming Orders 2+Klein Inventory Adjust Rate 2("Klein Supply-Line Accounting 2"*
**(Klein Desired Supply Line 2-Kleinmuntz Perceived Supplyline 2)-Klein Inventory Discrepancy 2*
))
Units: Units/Turn

(295) Kleinmuntz Ordering 3=
MAX(0, Klein Incoming Orders 3+Klein Inventory Adjust Rate 3("Klein Supply-Line Accounting 3"*
**(Klein Desired Supply Line 3-Kleinmuntz Perceived Supplyline 3)-Klein Inventory Discrepancy 3*
))
Units: Units/Turn

(296) Kleinmuntz Out=
DELAY FIXED(Change in Kleinmuntz Supply, 1 , Change in Kleinmuntz Supply)
Units: Units/Turn

(297) Kleinmuntz Out 0=
DELAY FIXED(Change in Kleinmuntz Supply 0, 1 , Change in Kleinmuntz Supply 0
)
Units: Units/Turn

(298) Kleinmuntz Out 1=
DELAY FIXED(Change in Kleinmuntz Supply 1, 1 , Change in Kleinmuntz Supply 1
)
Units: Units/Turn

(299) Kleinmuntz Out 2=
DELAY FIXED(Change in Kleinmuntz Supply 2, 1 , Change in Kleinmuntz Supply 2
)
Units: Units/Turn

(300) Kleinmuntz Out 3=
DELAY FIXED(Change in Kleinmuntz Supply 3, 1 , Change in Kleinmuntz Supply 3
)
Units: Units/Turn

(301) Kleinmuntz Perceived Supplyline= INTEG (
Change in Kleinmuntz Supply-Kleinmuntz Out,
Klein Desired Supply Line)
Units: Units

(302) Kleinmuntz Perceived Supplyline 0= INTEG (
Change in Kleinmuntz Supply 0-Kleinmuntz Out 0,
0)
Units: Units

(303) Kleinmuntz Perceived Supplyline 1= INTEG (
Change in Kleinmuntz Supply 1-Kleinmuntz Out 1,
0)
Units: Units

(304) Kleinmuntz Perceived Supplyline 2= INTEG (
Change in Kleinmuntz Supply 2-Kleinmuntz Out 2,
0)
Units: Units

(305) Kleinmuntz Perceived Supplyline 3= INTEG (
Change in Kleinmuntz Supply 3-Kleinmuntz Out 3,

0)
Units: Units

(306) Klien effective Inventory=
Kleinmuntz Inventory-Klein Backlog
Units: Units

(307) Klien effective Inventory 0=
Kleinmuntz Inventory 0-Klein Backlog 0
Units: Units

(308) Klien effective Inventory 1=
Kleinmuntz Inventory 1-Klein Backlog 1
Units: Units

(309) Klien effective Inventory 2=
Kleinmuntz Inventory 2-Klein Backlog 2
Units: Units

(310) Klien effective Inventory 3=
Kleinmuntz Inventory 3-Klein Backlog 3
Units: Units

(311) Margin of Safety=
0.5
Units: 1

(312) Margin of Safety 0=
0.5
Units: 1

(313) Margin of Safety 1=
0.5
Units: 1

(314) Margin of Safety 2=
0.5
Units: 1

(315) Margin of Safety 3=
0.5
Units: 1

(316) Memory Loss Time=
4
Units: Turn

(317) Memory Loss Time 0=
4
Units: Turn

(318) Memory Loss Time 1=
4
Units: Turn

(319) Memory Loss Time 2=
1
Units: Turn

(320) Memory Loss Time 3=
4
Units: Turn

(321) Order change Rate=
1
Units: Turn

(322) Order change Rate 0=
1
Units: Turn

(323) Order change Rate 1=
1
Units: Turn

(324) Order change Rate 2=
1
Units: Turn

(325) Order change Rate 3=
1
Units: Turn

(326) Outgoing Anchor=
MAX(0, MIN(Anchor Inventory/Time to Send 0 1, (Anchor Backlog/Time to Send 0 1
+Anchor Incoming Orders)))
Units: Units/Turn

(327) Outgoing Anchor 0=
MAX(0, MIN(Anchor Inventory 0/Time to Send 0 1 1, (Anchor Backlog 0/Time to Send 0
1 1
+Anchor Incoming Orders 0)))
Units: Units/Turn

(328) Outgoing Anchor 1=
MAX(0, MIN(Anchor Inventory 1/Time to Send 0 1 2, (Anchor Backlog 1/Time to Send 0 1 2
+Anchor Incoming Orders 1)))
Units: Units/Turn

(329) Outgoing Anchor 2=
MAX(0, MIN(Anchor Inventory 2/Time to Send 0 1 3, (Anchor Backlog 2/Time to Send 0 1 3
+Anchor Incoming Orders 2)))
Units: Units/Turn

(330) Outgoing Anchor 3=
MAX(0, MIN(Anchor Inventory 3/Time to Send 0 1 4, (Anchor Backlog 3/Time to Send 0 1 4
+Anchor Incoming Orders 3)))
Units: Units/Turn

(331) Outgoing Anchor Complex=
MAX(0, MIN(Anchor Complex Inventory/Time to Send 0 1 0, (Anchor Complex Backlog
/Time to Send 0 1 0+Anchor Complex Incoming Orders)))
Units: Units/Turn

(332) Outgoing Anchor Complex 0=
MAX(0, MIN(Anchor Complex Inventory 0/Time to Send 0 1 0 0, (Anchor Complex Backlog 0
/Time to Send 0 1 0 0+Anchor Complex Incoming Orders 0)))
Units: Units/Turn

(333) Outgoing Anchor Complex 1=
MAX(0, MIN(Anchor Complex Inventory 1/Time to Send 0 1 0 1, (Anchor Complex Backlog 1
/Time to Send 0 1 0 1+Anchor Complex Incoming Orders 1)))
Units: Units/Turn

(334) Outgoing Anchor Complex 2=
MAX(0, MIN(Anchor Complex Inventory 2/Time to Send 0 1 0 2, (Anchor Complex Backlog 2
/Time to Send 0 1 0 2+Anchor Complex Incoming Orders 2)))
Units: Units/Turn

(335) Outgoing Anchor Complex 3=
MAX(0, MIN(Anchor Complex Inventory 3/Time to Send 0 1 0 3, (Anchor Complex Backlog 3
/Time to Send 0 1 0 3+Anchor Complex Incoming Orders 3)))
Units: Units/Turn

(336) Outgoing Klien=
MAX(0, MIN(Kleinmuntz Inventory/Time to Send 0 0, (Klein Backlog/Time to Send 0 0 +Klein Incoming Orders)))
Units: Units/Turn

(337) Outgoing Klien 0=
MAX(0, MIN(Kleinmuntz Inventory 0/Time to Send 0 0 0, (Klein Backlog 0/Time to Send 0 0 0
+Klein Incoming Orders 0)))
Units: Units/Turn

(338) Outgoing Klien 1=
MAX(0, MIN(Kleinmuntz Inventory 1/Time to Send 0 0 1, (Klein Backlog 1/Time to Send 0 0 1
+Klein Incoming Orders 1)))
Units: Units/Turn

(339) Outgoing Klien 2=
MAX(0, MIN(Kleinmuntz Inventory 2/Time to Send 0 0 2, (Klein Backlog 2/Time to Send 0 0 2
+Klein Incoming Orders 2)))
Units: Units/Turn

(340) Outgoing Klien 3=
MAX(0, MIN(Kleinmuntz Inventory 3/Time to Send 0 0 3, (Klein Backlog 3/Time to Send 0 0 3
+Klein Incoming Orders 3)))
Units: Units/Turn

(341) Outgoing Safe=
MAX(0, MIN(Safe Inventory/Time to Send, (Safe Backlog/Time to Send+Safe Incoming Orders
)))
Units: Units/Turn

(342) Outgoing Safe 0=
MAX(0, MIN(Safe Inventory 0/Time to Send 1, (Safe Backlog 0/Time to Send 1
+Safe Incoming Orders 0)))
Units: Units/Turn

(343) Outgoing Safe 1=
MAX(0, MIN(Safe Inventory 1/Time to Send 2, (Safe Backlog 1/Time to Send 2
+Safe Incoming Orders 1)))
Units: Units/Turn

(344) Outgoing Safe 2=
MAX(0, MIN(Safe Inventory 2/Time to Send 3, (Safe Backlog 2/Time to Send 3

+Safe Incoming Orders 2)))
Units: Units/Turn

(345) Outgoing Safe 3=
MAX(0, MIN(Safe Inventory 3/Time to Send 4, (Safe Backlog 3/Time to Send 4
+Safe Incoming Orders 3)))
Units: Units/Turn

(346) Outgoing Sterman=
MAX(0, MIN(Sterman Inventory/Time to Send 0, (Sterman Backlog/Time to Send 0
+Sterman Incoming Orders)))
Units: Units/Turn

(347) Outgoing Sterman 0=
MAX(0, MIN(Sterman Inventory 0/Time to Send 0 2, (Sterman Backlog 0/Time to Send 0
2
+Sterman Incoming Orders 0)))
Units: Units/Turn

(348) Outgoing Sterman 1=
MAX(0, MIN(Sterman Inventory 1/Time to Send 0 3, (Sterman Backlog 1/Time to Send 0
3
+Sterman Incoming Orders 1)))
Units: Units/Turn

(349) Outgoing Sterman 2=
MAX(0, MIN(Sterman Inventory 2/Time to Send 0 4, (Sterman Backlog 2/Time to Send 0
4
+Sterman Incoming Orders 2)))
Units: Units/Turn

(350) Outgoing Sterman 3=
MAX(0, MIN(Sterman Inventory 3/Time to Send 0 5, (Sterman Backlog 3/Time to Send 0
5
+Sterman Incoming Orders 3)))
Units: Units/Turn

(351) Perceived Change in Inventory= INTEG (
Change in Perceived change in inventory,
0)
Units: Units/Turn

(352) Perceived Change in Inventory 0= INTEG (
Change in Perceived change in inventory 0,
0)
Units: Units/Turn

(353) Perceived Change in Inventory 1= INTEG (
Change in Perceived change in inventory 1,
0)
Units: Units/Turn

(354) Perceived Change in Inventory 2= INTEG (
Change in Perceived change in inventory 2,
0)
Units: Units/Turn

(355) Perceived Change in Inventory 3= INTEG (
Change in Perceived change in inventory 3,
0)
Units: Units/Turn

(356) Perceived Effective Inventory= INTEG (
Change in Perceived Inventory,
12)
Units: Units

(357) Perceived Effective Inventory 0= INTEG (
Change in Perceived Inventory 0,
12)
Units: Units

(358) Perceived Effective Inventory 1= INTEG (
Change in Perceived Inventory 1,
12)
Units: Units

(359) Perceived Effective Inventory 2= INTEG (
Change in Perceived Inventory 2,
12)
Units: Units

(360) Perceived Effective Inventory 3= INTEG (
Change in Perceived Inventory 3,
12)
Units: Units

(361) Rate of Return=
4
Units: Turns

(362) Rate of Return 0=
1
Units: Turns

(363) Rate of Return 0 0=
1
Units: Turns

(364) Rate of Return 0 1=
1
Units: Turns

(365) Rate of Return 0 2=
1
Units: Turns

(366) Rate of Return 0 3=
1
Units: Turns

(367) Rate of Return 1=
4
Units: Turns

(368) Rate of Return 1 0=
4
Units: Turns

(369) Rate of Return 1 0 0=
4
Units: Turns

(370) Rate of Return 1 0 1=
4
Units: Turns

(371) Rate of Return 1 0 2=
4
Units: Turns

(372) Rate of Return 1 0 3=
4
Units: Turns

(373) Rate of Return 2=
4
Units: Turns

(374) Rate of Return 3=
4

Units: Turns

(375) Rate of Return 4=
4
Units: Turns

(376) Rate of Return 5=
2
Units: Turns

(377) Retailer Effective Inventory=
Retailer Inventory-Backlog Retailer
Units: Units

(378) Retailer Effective Inventory 0=
Retailer Inventory 0-Backlog Retailer 0
Units: Units

(379) Retailer Effective Inventory 1=
Retailer Inventory 1-Backlog Retailer 1
Units: Units

(380) Retailer Effective Inventory 2=
Retailer Inventory 2-Backlog Retailer 2
Units: Units

(381) Retailer Effective Inventory 3=
Retailer Inventory 3-Backlog Retailer 3
Units: Units

(382) Retailer Inventory= INTEG (
-Shipments from Retailer+Shipments to retailer,
12)
Units: Units

(383) Retailer Inventory 0= INTEG (
-Shipments from Retailer 0+Shipments to retailer 0,
12)
Units: Units

(384) Retailer Inventory 1= INTEG (
-Shipments from Retailer 1+Shipments to retailer 1,
12)
Units: Units

(385) Retailer Inventory 2= INTEG (
-Shipments from Retailer 2+Shipments to retailer 2,

12)
Units: Units

(386) Retailer Inventory 3= INTEG (
-Shipments from Retailer 3+Shipments to retailer 3,
12)
Units: Units

(387) Retailer Orders=
MAX(0, Customer Orders+(Backlog Retailer+Desired Inventory-Retailer Inventory
*)*Inventory Adjustment Factor)*
Units: Units/Turn

(388) Retailer Orders 0=
MAX(0, Customer Orders 0+(Backlog Retailer 0+Desired Inventory 0-Retailer
Inventory 0
*)*Inventory Adjustment Factor 0)*
Units: Units/Turn

(389) Retailer Orders 1=
MAX(0, Customer Orders 1+(Backlog Retailer 1+Desired Inventory 1-Retailer
Inventory 1
*)*Inventory Adjustment Factor 1)*
Units: Units/Turn

(390) Retailer Orders 2=
MAX(0, Customer Orders 2+(Backlog Retailer 2+Desired Inventory 2-Retailer
Inventory 2
*)*Inventory Adjustment Factor 2)*
Units: Units/Turn

(391) Retailer Orders 3=
MAX(0, Customer Orders 3+(Backlog Retailer 3+Desired Inventory 3-Retailer
Inventory 3
*)*Inventory Adjustment Factor 3)*
Units: Units/Turn

(392) Safe Backlog= INTEG (
Safe Backlog Change,
0)
Units: Units

(393) Safe Backlog 0= INTEG (
Safe Backlog Change 0,
0)
Units: Units

```
(394)  Safe Backlog 1= INTEG (
               Safe Backlog Change 1,
                           0)
       Units: Units

(395)  Safe Backlog 2= INTEG (
               Safe Backlog Change 2,
                           0)
       Units: Units

(396)  Safe Backlog 3= INTEG (
               Safe Backlog Change 3,
                           0)
       Units: Units

(397)  Safe Backlog Change=
               Safe Incoming Orders-Outgoing Safe
       Units: Units/Turn

(398)  Safe Backlog Change 0=
               Safe Incoming Orders 0-Outgoing Safe 0
       Units: Units/Turn

(399)  Safe Backlog Change 1=
               Safe Incoming Orders 1-Outgoing Safe 1
       Units: Units/Turn

(400)  Safe Backlog Change 2=
               Safe Incoming Orders 2-Outgoing Safe 2
       Units: Units/Turn

(401)  Safe Backlog Change 3=
               Safe Incoming Orders 3-Outgoing Safe 3
       Units: Units/Turn

(402)  Safe Desired Inventory=
               12
       Units: Units

(403)  Safe Desired Inventory 0=
               12
       Units: Units

(404)  Safe Desired Inventory 1=
               12
       Units: Units
```

(405) Safe Desired Inventory 2=
12
Units: Units

(406) Safe Desired Inventory 3=
12
Units: Units

(407) Safe effective Inventory=
Safe Inventory-Safe Backlog
Units: Units

(408) Safe effective Inventory 0=
Safe Inventory 0-Safe Backlog 0
Units: Units

(409) Safe effective Inventory 1=
Safe Inventory 1-Safe Backlog 1
Units: Units

(410) Safe effective Inventory 2=
Safe Inventory 2-Safe Backlog 2
Units: Units

(411) Safe effective Inventory 3=
Safe Inventory 3-Safe Backlog 3
Units: Units

(412) Safe Incoming Orders=
4+STEP(4, 4)
Units: Units/Turn

(413) Safe Incoming Orders 0=
4+STEP(4, 4)
Units: Units/Turn

(414) Safe Incoming Orders 1=
DELAY FIXED(Safe Outgoing Orders 0, 2, 4)
Units: Units/Turn

(415) Safe Incoming Orders 2=
DELAY FIXED(Safe Outgoing Orders 1, 2, 4)
Units: Units/Turn

(416) Safe Incoming Orders 3=
DELAY FIXED(Safe Outgoing Orders 2, 2, 4)
Units: Units/Turn

(417) Safe Incoming Shipments=
Safe Outgoing Orders
Units: Units/Turn

(418) Safe Incoming Shipments 0=
Outgoing Safe 1
Units: Units/Turn

(419) Safe Incoming Shipments 1=
Outgoing Safe 2
Units: Units/Turn

(420) Safe Incoming Shipments 2=
Outgoing Safe 3
Units: Units/Turn

(421) Safe Incoming Shipments 3=
Safe Outgoing Orders 3
Units: Units/Turn

(422) Safe Inventory= INTEG (
Incoming Safe-Outgoing Safe,
12)
Units: Units

(423) Safe Inventory 0= INTEG (
Incoming Safe 0-Outgoing Safe 0,
12)
Units: Units

(424) Safe Inventory 1= INTEG (
Incoming Safe 1-Outgoing Safe 1,
12)
Units: Units

(425) Safe Inventory 2= INTEG (
Incoming Safe 2-Outgoing Safe 2,
12)
Units: Units

(426) Safe Inventory 3= INTEG (
Incoming Safe 3-Outgoing Safe 3,
12)
Units: Units

(427) Safe Inventory Discrepancy=

Safe Inventory-Safe Desired Inventory
Units: Units

(428) Safe Inventory Discrepancy 0=
Safe Inventory 0-Safe Desired Inventory 0
Units: Units

(429) Safe Inventory Discrepancy 1=
Safe Inventory 1-Safe Desired Inventory 1
Units: Units

(430) Safe Inventory Discrepancy 2=
Safe Inventory 2-Safe Desired Inventory 2
Units: Units

(431) Safe Inventory Discrepancy 3=
Safe Inventory 3-Safe Desired Inventory 3
Units: Units

(432) Safe Outgoing Orders=
*MAX(0, Safe Incoming Orders-Margin of Safety*Perceived Change in Inventory*
*-Safe Inventory Discrepancy*Inventory Correction Rate*
)
Units: Units/Turn

(433) Safe Outgoing Orders 0=
*MAX(0, Safe Incoming Orders 0-Margin of Safety 0*Perceived Change in Inventory 0*
*-Safe Inventory Discrepancy 0*Inventory Correction Rate 0*
)
Units: Units/Turn

(434) Safe Outgoing Orders 1=
*MAX(0, Safe Incoming Orders 1-Margin of Safety 1*Perceived Change in Inventory 1*
*-Safe Inventory Discrepancy 1*Inventory Correction Rate 1*
)
Units: Units/Turn

(435) Safe Outgoing Orders 2=
*MAX(0, Safe Incoming Orders 2-Margin of Safety 2*Perceived Change in Inventory 2*
*-Safe Inventory Discrepancy 2*Inventory Correction Rate 2*
)
Units: Units/Turn

(436) Safe Outgoing Orders 3=
*MAX(0, Safe Incoming Orders 3-Margin of Safety 3*Perceived Change in Inventory 3*
*-Safe Inventory Discrepancy 3*Inventory Correction Rate 3*
)

Units: Units/Turn

(437) SAVEPER =
TIME STEP
Units: Turns [0,?]
The frequency with which output is stored.

(438) Shipments from Retailer=
MAX(0, MIN(Retailer Inventory/Time to Ship Inventory, (Backlog Retailer/Time to Ship Inventory
+Customer Orders)))
Units: Units/Turn

(439) Shipments from Retailer 0=
MAX(0, MIN(Retailer Inventory 0/Time to Ship Inventory 0, (Backlog Retailer 0
/Time to Ship Inventory 0+Customer Orders 0)))
Units: Units/Turn

(440) Shipments from Retailer 1=
MAX(0, MIN(Retailer Inventory 1/Time to Ship Inventory 1, (Backlog Retailer 1
/Time to Ship Inventory 1+Customer Orders 1)))
Units: Units/Turn

(441) Shipments from Retailer 2=
MAX(0, MIN(Retailer Inventory 2/Time to Ship Inventory 2, (Backlog Retailer 2
/Time to Ship Inventory 2+Customer Orders 2)))
Units: Units/Turn

(442) Shipments from Retailer 3=
MAX(0, MIN(Retailer Inventory 3/Time to Ship Inventory 3, (Backlog Retailer 3
/Time to Ship Inventory 3+Customer Orders 3)))
Units: Units/Turn

(443) Shipments to retailer=
DELAY FIXED(Shipments from Retailer 0, 2, Shipments from Retailer 0)
Units: Units/Turn

(444) Shipments to retailer 0=
DELAY FIXED(Shipments from Retailer 1, 2, Shipments from Retailer 1)
Units: Units/Turn

(445) Shipments to retailer 1=
DELAY FIXED(Shipments from Retailer 2, 2, Shipments from Retailer 2)
Units: Units/Turn

(446) Shipments to retailer 2=
DELAY FIXED(Retailer Orders 2, 2, Retailer Orders 2)

```
        Units: Units/Turn

(447)   Shipments to retailer 3=
                DELAY FIXED( Retailer Orders 3, 4, Retailer Orders 3)
        Units: Units/Turn

(448)   Sterman Backlog= INTEG (
                Sterman Backlog Change,
                        0)
        Units: Units

(449)   Sterman Backlog 0= INTEG (
                Sterman Backlog Change 0,
                        0)
        Units: Units

(450)   Sterman Backlog 1= INTEG (
                Sterman Backlog Change 1,
                        0)
        Units: Units

(451)   Sterman Backlog 2= INTEG (
                Sterman Backlog Change 2,
                        0)
        Units: Units

(452)   Sterman Backlog 3= INTEG (
                Sterman Backlog Change 3,
                        0)
        Units: Units

(453)   Sterman Backlog Change=
                Sterman Incoming Orders-Outgoing Sterman
        Units: Units/Turn

(454)   Sterman Backlog Change 0=
                Sterman Incoming Orders 0-Outgoing Sterman 0
        Units: Units/Turn

(455)   Sterman Backlog Change 1=
                Sterman Incoming Orders 1-Outgoing Sterman 1
        Units: Units/Turn

(456)   Sterman Backlog Change 2=
                Sterman Incoming Orders 2-Outgoing Sterman 2
        Units: Units/Turn
```

(457) Sterman Backlog Change 3=
Sterman Incoming Orders 3-Outgoing Sterman 3
Units: Units/Turn

(458) Sterman Desired Inventory=
12
Units: Units

(459) Sterman Desired Inventory 0=
12
Units: Units

(460) Sterman Desired Inventory 1=
12
Units: Units

(461) Sterman Desired Inventory 2=
12
Units: Units

(462) Sterman Desired Inventory 3=
12
Units: Units

(463) Sterman Desired Supply Line=
*Sterman Incoming Orders*Rate of Return*
Units: Units

(464) Sterman Desired Supply Line 0=
*Sterman Incoming Orders 0*Rate of Return 2*
Units: Units

(465) Sterman Desired Supply Line 1=
*Sterman Incoming Orders 1*Rate of Return 3*
Units: Units

(466) Sterman Desired Supply Line 2=
*Sterman Incoming Orders 2*Rate of Return 4*
Units: Units

(467) Sterman Desired Supply Line 3=
*Sterman Incoming Orders 3*Rate of Return 5*
Units: Units

(468) Sterman effective Inventory=
Sterman Inventory-Sterman Backlog
Units: Units

(469) Sterman effective Inventory 0=
Sterman Inventory 0-Sterman Backlog 0
Units: Units

(470) Sterman effective Inventory 1=
Sterman Inventory 1-Sterman Backlog 1
Units: Units

(471) Sterman effective Inventory 2=
Sterman Inventory 2-Sterman Backlog 2
Units: Units

(472) Sterman effective Inventory 3=
Sterman Inventory 3-Sterman Backlog 3
Units: Units

(473) Sterman Incoming Orders=
4+STEP(4, 4)
Units: Units/Turn

(474) Sterman Incoming Orders 0=
4+STEP(4, 4)
Units: Units/Turn

(475) Sterman Incoming Orders 1=
DELAY FIXED(Sterman Ordering 0, 2, 4)
Units: Units/Turn

(476) Sterman Incoming Orders 2=
DELAY FIXED(Sterman Ordering 1, 2, 4)
Units: Units/Turn

(477) Sterman Incoming Orders 3=
DELAY FIXED(Sterman Ordering 2, 2, 4)
Units: Units/Turn

(478) Sterman Incoming Shipments=
Sterman Ordering
Units: Units/Turn

(479) Sterman Incoming Shipments 0=
Outgoing Sterman 1
Units: Units/Turn

(480) Sterman Incoming Shipments 1=
Outgoing Sterman 2

```
        Units: Units/Turn

(481)   Sterman Incoming Shipments 2=
                Outgoing Sterman 3
        Units: Units/Turn

(482)   Sterman Incoming Shipments 3=
                Sterman Ordering 3
        Units: Units/Turn

(483)   Sterman Inventory= INTEG (
                Incoming Sterman-Outgoing Sterman,
                        12)
        Units: Units

(484)   Sterman Inventory 0= INTEG (
                Incoming Sterman 0-Outgoing Sterman 0,
                        12)
        Units: Units

(485)   Sterman Inventory 1= INTEG (
                Incoming Sterman 1-Outgoing Sterman 1,
                        12)
        Units: Units

(486)   Sterman Inventory 2= INTEG (
                Incoming Sterman 2-Outgoing Sterman 2,
                        12)
        Units: Units

(487)   Sterman Inventory 3= INTEG (
                Incoming Sterman 3-Outgoing Sterman 3,
                        12)
        Units: Units

(488)   Sterman Inventory Adjust Rate=
                1
        Units: 1/Turn

(489)   Sterman Inventory Adjust Rate 0=
                1
        Units: 1/Turn

(490)   Sterman Inventory Adjust Rate 1=
                1
        Units: 1/Turn
```

(491) Sterman Inventory Adjust Rate 2=
1
Units: 1/Turn

(492) Sterman Inventory Adjust Rate 3=
1
Units: 1/Turn

(493) Sterman Inventory Discrepancy=
Sterman effective Inventory-Sterman Desired Inventory
Units: Units

(494) Sterman Inventory Discrepancy 0=
Sterman effective Inventory 0-Sterman Desired Inventory 0
Units: Units

(495) Sterman Inventory Discrepancy 1=
Sterman effective Inventory 1-Sterman Desired Inventory 1
Units: Units

(496) Sterman Inventory Discrepancy 2=
Sterman effective Inventory 2-Sterman Desired Inventory 2
Units: Units

(497) Sterman Inventory Discrepancy 3=
Sterman effective Inventory 3-Sterman Desired Inventory 3
Units: Units

(498) Sterman Ordering=
Sterman Incoming Orders+Sterman Inventory Adjust Rate("Sterman Supply-Line Accounting"*
**(Sterman Desired Supply Line-Supply Line)-Sterman Inventory Discrepancy)*
Units: Units/Turn

(499) Sterman Ordering 0=
MAX(0, (Sterman Incoming Orders 0+Sterman Inventory Adjust Rate 0("Sterman Supply-Line Accounting 0"*
**(Sterman Desired Supply Line 0-Supply Line 0)-Sterman Inventory Discrepancy 0*
)))
Units: Units/Turn

(500) Sterman Ordering 1=
MAX(0, (Sterman Incoming Orders 1+Sterman Inventory Adjust Rate 1("Sterman Supply-Line Accounting 1"*
**(Sterman Desired Supply Line 1-Supply Line 1)-Sterman Inventory Discrepancy 1*
)))
Units: Units/Turn

(501) Sterman Ordering 2=
MAX(0, (Sterman Incoming Orders 2+Sterman Inventory Adjust Rate 2("Sterman Supply-Line Accounting 2"*
**(Sterman Desired Supply Line 2-Supply Line 2)-Sterman Inventory Discrepancy 2*
)))
Units: Units/Turn

(502) Sterman Ordering 3=
MAX(0, (Sterman Incoming Orders 3+Sterman Inventory Adjust Rate 3("Sterman Supply-Line Accounting 3"*
**(Sterman Desired Supply Line 3-Supply Line 3)-Sterman Inventory Discrepancy 3*
)))
Units: Units/Turn

(503) "Sterman Supply-Line Accounting 0"=
0.5
Units: 1

(504) "Sterman Supply-Line Accounting 1"=
0.5
Units: 1

(505) "Sterman Supply-Line Accounting 2"=
0.5
Units: 1

(506) "Sterman Supply-Line Accounting 3"=
0.5
Units: 1

(507) "Sterman Supply-Line Accounting"=
0.5
Units: 1

(508) Supply Line= INTEG (
Change in Supply Line,
16)
Units: Units

(509) Supply Line 0= INTEG (
Change in Supply Line 0,
Sterman Desired Supply Line 0)
Units: Units

(510) Supply Line 1= INTEG (
Change in Supply Line 1,

```
                        Sterman Desired Supply Line 1)
        Units: Units

(511)   Supply Line 2= INTEG (
                Change in Supply Line 2,
                        Sterman Desired Supply Line 2)
        Units: Units

(512)   Supply Line 3= INTEG (
                Change in Supply Line 3,
                        Sterman Desired Supply Line 3)
        Units: Units

(513)   TIME STEP  = 0.125
        Units: Turns [0,?]
        The time step for the simulation.

(514)   Time to Change=
                1
        Units: Turn

(515)   Time to Change 0=
                1
        Units: Turn

(516)   Time to Change 1=
                1
        Units: Turn

(517)   Time to Change 2=
                1
        Units: Turn

(518)   Time to Change 3=
                1
        Units: Turn

(519)   Time to change view=
                1
        Units: Turn

(520)   Time to change view 0=
                1
        Units: Turn

(521)   Time to change view 1=
                1
```

Units: Turn

(522) Time to change view 2=
1
Units: Turn

(523) Time to change view 3=
1
Units: Turn

(524) Time to Send=
1
Units: Turn

(525) Time to Send 0=
1
Units: Turn

(526) Time to Send 0 0=
1
Units: Turn

(527) Time to Send 0 0 0=
1
Units: Turn

(528) Time to Send 0 0 1=
1
Units: Turn

(529) Time to Send 0 0 2=
1
Units: Turn

(530) Time to Send 0 0 3=
1
Units: Turn

(531) Time to Send 0 1=
1
Units: Turn

(532) Time to Send 0 1 0=
1
Units: Turn

(533) Time to Send 0 1 0 0=

1
Units: Turn

(534) Time to Send 0 1 0 1=
1
Units: Turn

(535) Time to Send 0 1 0 2=
1
Units: Turn

(536) Time to Send 0 1 0 3=
1
Units: Turn

(537) Time to Send 0 1 1=
1
Units: Turn

(538) Time to Send 0 1 2=
1
Units: Turn

(539) Time to Send 0 1 3=
1
Units: Turn

(540) Time to Send 0 1 4=
1
Units: Turn

(541) Time to Send 0 2=
1
Units: Turn

(542) Time to Send 0 3=
1
Units: Turn

(543) Time to Send 0 4=
1
Units: Turn

(544) Time to Send 0 5=
1
Units: Turn

(545) Time to Send 1=
1
Units: Turn

(546) Time to Send 2=
1
Units: Turn

(547) Time to Send 3=
1
Units: Turn

(548) Time to Send 4=
1
Units: Turn

(549) Time to Ship Inventory=
1
Units: Turn

(550) Time to Ship Inventory 0=
1
Units: Turn

(551) Time to Ship Inventory 1=
1
Units: Turn

(552) Time to Ship Inventory 2=
1
Units: Turn

(553) Time to Ship Inventory 3=
1
Units: Turn