4.2 Desktop Modeling and Simulation: Parsimonious, Yet Effective Discrete-Event Simulation Analysis

Desktop Modeling and Simulation: Parsimonious, Yet Effective Discrete-Event Simulation Analysis

James R. Bradley
Mason School of Business
College of William and Mary
james.bradley@mason.wm.edu

This paper evaluates how quickly students can be trained to construct useful discrete-event simulation models using Excel. The typical supply chain used by many large national retailers is described, and an Excel-based simulation model is constructed of it. The set of programming and simulation skills required for development of that model are then determined: we conclude that six hours of training are required to teach the skills to MBA students. The simulation presented here contains all fundamental functionality of a simulation model, and so our result holds for any discrete-event simulation model. We argue, therefore, that industry workers with the same technical skill set as students having completed one year in an MBA program can be quickly trained to construct simulation models. This result gives credence to the efficacy of *Desktop Modeling and Simulation* whereby simulation analyses can be quickly developed, run, and analyzed with widely available software, namely Excel.

1.0 INTRODUCTION

Simulation analyses in business typically use one of two simulation methodologies, Monte Carlo simulation or discrete-event simulation. These analyses can be conducted with many different types of software platforms, including (1) commercial discrete-event and Monte Carlo software packages, and (2) custom programs written in Visual Basic, C#, Java, or other similar programming language. Advantages of the first approach include avoiding the need to program the functionality that comes with a commercially available product, such as the generation of random variables and, for discrete-event simulation, graphic user interfaces that make constructing a model more intuitive. One disadvantage of using an existing simulation program is their cost, which can be thousands of dollars. Advantages of authoring one's own simulation program is that it is likely to run much faster than commercial packages and custom features can be created that are not available in packaged software. Disadvantages of custom simulation models are acquiring the requisite programming skills and the time required for programming. Also, programming animation, if it is desired, can add considerably to development time.

In this paper, we consider an alternative to these two "endpoints" on a continuum of software implementations of simulation. Specifically, we consider developing discrete-event software analyses in Excel, supplemented by simple Visual Basic for Applications (VBA) code. (While we focus on discrete-event simulation in this paper, our conclusion is likely to be applicable also to Monte Carlo simulation as is investigated by Guerrero (2010b).) We show that constructing a complex simulation with this ubiquitous software package can be accomplished with only minimal training in VBA. This approach mitigates many of the disadvantages of the two endpoint alternatives. First, Excel is inexpensive, since virtually everybody has access to it on their personal computer. Second, although the VBA knowledge required is minimal compared to that required to generate custom programs, many users would find Excel/VBA to be not much more difficult, if at all, that learning the graphical user interface for simulation software platforms. Third, while animation of processes is sometimes useful, most often business decisions are driven the simulation data rather than animation. Animation is, by the way, possible in Excel although it requires additional VBA programming knowledge.

This paper supports the foregoing assertions about discrete-event simulations by showing an example simulation of a supply chain where decisions are made

regarding replenishment of goods from an overseas source. This model is coded in the simplest manner possible in order to determine the minimum possible training to enable a user to construct a valid simulation. Our focus, then, is on how the simulation is implemented rather than on analyzing the simulation results. We find that Excel can be used to quickly construct such models, which provides a basis of support for a trend in simulation, which has been called "Desktop Modeling and Simulation" by Guerrero (2010b). The hallmarks of desktop modeling and simulation are that small models can be quickly constructed in business by individuals with a minimum of training to make effective decisions on possibly complex business scenarios.

2.0 BODY

2.1 Simulation Context

The context for the sample discrete-event model used in this paper is a global supply chain, which is the type that might be operated by a large national retailer (e.g., Wal-Mart, Target). The supply chain has five links (see Figure 1):

1. Overseas source of goods
2. Crossdocking facility
3. Import distribution Center (IDC)
4. Regional Distribution Centers (RDCs)
5. Retail Stores

The first two links are connected, primarily, by oceangoing container ships and logistics at the originating port. The remaining links are in the US and are linked in this simulation model by truck transportation. (In other contexts, rail or barge transport might be involved.)

Quickly described, each of these links serves the following purposes: the overseas source can be a manufacturer or distributor of goods that might possibly go through a consolidation step before arriving at the port of origin; the Crossdocking facility unloads the incoming containers, which may contain only a small number of stock keeping units (SKUs) and allocates the large quantity of each SKU in each container into smaller quantities in many trailers destined for RDCs or retail stores; the remainder of the incoming inventory not allocated to RDCs is sent to the Import Distribution Center (IDC) for storage until such time as it is needed at the RDCs; and the Retail Stores which, obviously, make the goods available to customers.

Typically in such a supply chain, ships arrive at a port close to the IDC from each origin every week or so with a particular SKU whereas, in the intervening time, the inventories temporarily stored in the IDC can be used to replenish the RDCs and retail stores that experience heavy demand.

The purpose of the supply chain simulation is to determine decisions about:

1. Quantities of imported goods to immediately allocate to RDCs
2. Daily replenishment quantities from the IDC to the RDCs

Retaining inventory at the IDC until the need arises at the RDCs rather than sending it immediately to the RDCs is called *postponement*. This is an emerging supply chain management tactic that can reduce inventory levels. Alternately, if all the incoming inventories are immediately sent on to RDCs, then a company would be vulnerable to fluctuating demand; if demand at one RDC were higher than expected, which will invariably happen, then that RDC would run out of stock while inventory would be left over at other RDCs where demand is low. This situation is expensive to resolve (i.e., transshipment among RDCs can be used), or infeasible due to financial considerations. Thus, inventories are high (leftover inventory at some RDCs) while service to customer is low (at some RDCs that experience high demand). Waiting to observe actual demand before allocating

goods allows better placement of inventory, which can reduce investment in inventory and improve availability of goods to customers. Just how one should implement an appropriate postponement strategy, however, is difficult to analyze. Analysis with calculus to determine an optimal decision policy is intractable, and so simulation offers a viable means of studying such a system.

2.2 Simulation Model Description

The main worksheet in the simulation workbook is shown in Figure 2. Note that the sheet is designed in accordance with "spreadsheet feng shui" as advocated by Guerrero (2010a). Specifically, color coding denotes regions of the worksheet that serve different purposes (problem parameters, ordering decision policies, day-to-day orders resulting from ordering policies, daily inventory status at IDC and RDCs, etc.), which makes the simulation more readily understandable.

The two parameters of the ordering decision policy are two target service levels for the weekly replenishment from arriving imports and the daily replenishment from the IDC. The 60% service level in Figure 2 indicates that when an imported shipment arrives RDCs each receive enough inventory such that in 60% of the weeks they would not stock out. The daily replenishment target service level of 97% indicates that on a daily basis each RDC receives a shipment quantity that is sufficient such that 97% of the days they would not stock out. With both weekly and daily replenishment it is possible that either the incoming imported shipment or the inventory at the IDC (respectively) is sometimes not sufficient to reach these service level targets, in which case an alternate allocation scheme is used. This service policy is a modification of a well known optimal policy in the inventory management literature (i.e., the optimal solution to a multiple-newsvendor model with constrained supply).

The crux of the decision is to determine how much of the incoming imported goods to allocate immediately to RDCs: transportation cost can be saved if more inventory is sent immediately to the RDCs whereas, if too much inventory is sent, it is likely to be sent to the "wrong" RDC such that one RDC will run out of inventory while another RDC has leftover inventory.

Figure 3 shows how the resulting fill rate on the y-axis (the percentage of customers immediately satisfied) and the average weekly cost of inventory and shipping (x-axis) changes as the service level from inventory immediately allocated from imports increases from 10% to 80%. The simulation reveals that the fill rate, at first, increases as weekly target service level from imports increases, but fill rate decreases as the weekly service level increases past 40%. This is due to, as described above, the increasing frequency of one RDC having leftover inventory at the end of a week while other RDCs have run out of inventory.

2.3 Simulation Model Structure

The key components of a basic discrete event simulation algorithm are:

1. Generating random variables that represent uncertain events
2. Keeping track of the state of the system as the simulation progresses
3. Saving simulation data for later analysis
4. Displaying simulation results
5. Controlling how long the simulation is run

This simulation has been constructed with the intent of minimizing (i) the amount of knowledge one must have regarding the general methodology of simulation, and (ii) expertise in spreadsheet construction and VBA programming, which motivated the particular programming tack that we describe below. Spreadsheet simulation functionality can be implemented in many

different ways, for example, within the spreadsheet, in the VBA "behind" the spreadsheet, in an Excel add-in, or in another programming language which is compiled and linked to the spreadsheet. Our goal of minimizing the training required to implement simulation motivates us to avoid programmatic approaches where possible in favor of using the Excel spreadsheet itself because many more people are familiar with Excel than with computer programming languages.

Accordingly, rather than generating random variables with an algorithm coded in VBA or some other programming language, we use the spreadsheet in the simulation model. In the global supply chain spreadsheet, we model daily demand at the RDCs as being normally distributed. Thus, we use the RAND() spreadsheet function (which generates a random variable from the continuous uniform distribution) together with the NORMINV() spreadsheet function to "draw" random variables using the inverse probability function. In addition, we use the MAX() function to ensure that demand is not less than zero (which makes no sense in this context). Using a distribution's inverse is not always feasible. For example, for discrete probability distributions where Excel does not offer an inverse function, the target cumulative probability distribution can be computed in a range within a worksheet such that random variables can be generated using the RAND()and VLOOKUP() spreadsheet functions. The probability function range would include at a minimum the possible values of the random variable (in ascending order) and a column for the corresponding cumulative probability function. VLOOKUP() can then be used to identify the appropriate random variable value which could be found in the row where the cumulative probability first exceeds the RAND() value. A similar approach is possible using the MATCH() spreadsheet function. Implementing inverse probability functions in this manner within the spreadsheet is likely easier to teach to programming neophytes than a programmatic approach.

The state of the system at any point of time consists of the inventory levels at the IDC and at each RDC. Showing inventory levels visually on the worksheet is beneficial so that users can observe what is happening, but it requires VBA programming. Perhaps the simplest approach is to designate one cell in the simulation worksheet to contain each inventory quantity and then use the Range.Offset.Value VBA statement to update inventory levels at the end of each day. Inventory at the end of the day (the new value in the cell) is the old value in the cell plus the quantity of inventory received, minus the outgoing inventory. (Note that a simplistic approach using a worksheet function to compute this value without VBA results in a circular reference.) For example, the following statement is used to update the inventory count at each RDC based on the amount of inventory shipped from the IDC to the RDCs, where *shipQ* is a VBA variable that denotes the quantity shipped from the IDC to the RDC in question:

```
.Range("simInvRDC").Offset(RDC, 0).Value
= .Range("simInvRDC").Offset(RDC,
0).Value + shipQ
```

A similar statement is used later in the program to reduce the RDC inventory due to goods that are delivered to retail stores and taken out of inventory. Note that the VBA code makes extensive use of named ranges, such as "simInvRDC", to make VBA code more readable. If-then VBA statements are also required to account for inventory, as well as understanding the basic arithmetic operators in VBA (+, -, *, /).

In the case of the example simulation, storing inventory level history at each of the DCs is useful for subsequent analysis. Thus, we keep track of the IDC and RDC inventory in a "Results" worksheet. These data can then be graphed to better help understand the functioning of the supply chain over time under the proposed order

policy. Storing these data requires VBA programming, albeit perhaps the simplest statement to understand, namely some form of the aforementioned Range.Offset.Value command, which can be used to copy the current inventory levels from the simulation worksheet onto the appropriate row and column of the "Results" worksheet. When the range for the simulation output can be identified in advance, a graph can be constructed beforehand to display results automatically when the simulation finishes.

Controlling the number of weeks over which the simulation is run requires VBA also. In this case, we have used a For-Next loop in VBA to run the simulation for the predetermined number of weeks indicated on the main simulation worksheet.

Note that the simulation worksheet must be recalculated each day in order to refresh (generate) new demand at each RDC. This requires either the Worksheet.Calculate or the Range.Calculate VBA function because, as we describe below, it is normally wise to turn off the automatic workbook recalculation.

While the above functionality arguably constitutes the basic implementation of a simulation analysis, additionally, other knowledge is advantageous to improve the performance of the simulation. Specifically, to increase the speed at which the simulation runs, it is beneficial in Excel to suspend the graphical updating of the spreadsheet and, also, to suspend the recalculation of the spreadsheet, which in automatic mode occurs whenever the value in any cell changes. Otherwise, the simulation can be very slow. To disable these features, one must use these commands:

- Application.Calculation = xlCalculationManual
- Application.ScreenUpdating = False

It is good practice to turn screen updating and recalculation back on at the end of the simulation with these statements:

- Application.Calculation = xlCalculationAutomatic
- Application.ScreenUpdating = True

Also, while not strictly necessary, we have declared VBA variables in the simulation, which helps to avoid programming errors.

2.4 Required Ancillary Skills

In order to draw appropriate conclusions from a simulation analysis, one must understand hypothesis testing, or alternative statistical methods that can discern the significance of differences in sample statistics generated by simulation analyses. In the target population discussed earlier, MBA students have already been exposed to hypothesis testing by the time they encounter a supply chain management course, so this is a moot point. This topic would need to be covered for some other populations, however. In addition, for MBA students, and possibly all students, must be made aware of the Law of Large Numbers and the Central Limit Theorem in order to understand the necessity of sometimes increasing the runtime of simulations in order to obtain definitive results.

2.5 The "Next" Simulation Skills

We mention one other programming technique that is useful when the data generated by simulation analyses is large. That is, the data from simulation studies can be stored in a database rather than in a worksheet as described above. This is beneficial when the data are sizeable. VBA offers connectivity to Access, SQL Server, and other databases. One should take care, however, to avoid using Access when the data cause the database to exceed roughly 2GB in size. At that point Access fails and, in the author's experience, gives no clear error message to indicate the specifics of the error.

3.0 DISCUSSION

A summary of the knowledge required to implement the simulation model as described in Section 2.3 is as follows:

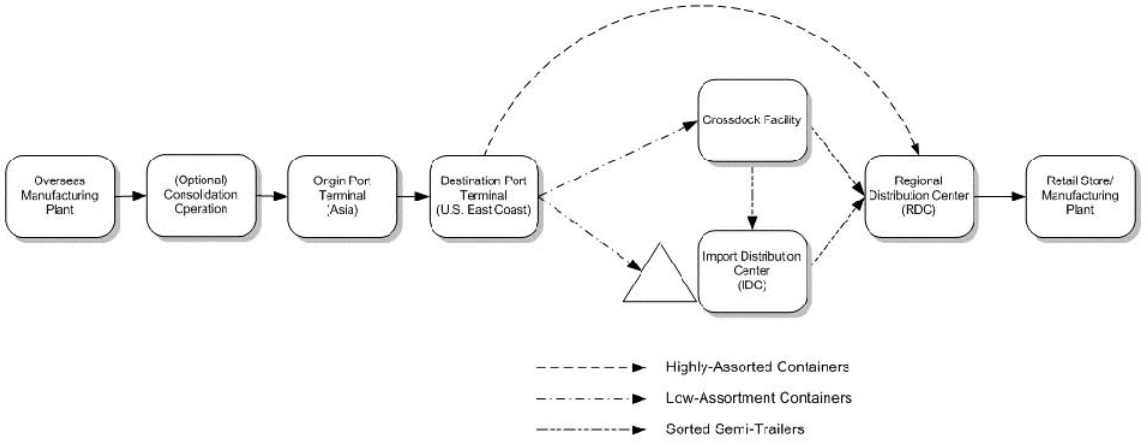

Figure 1: Diagram of Example Global Supply Chain

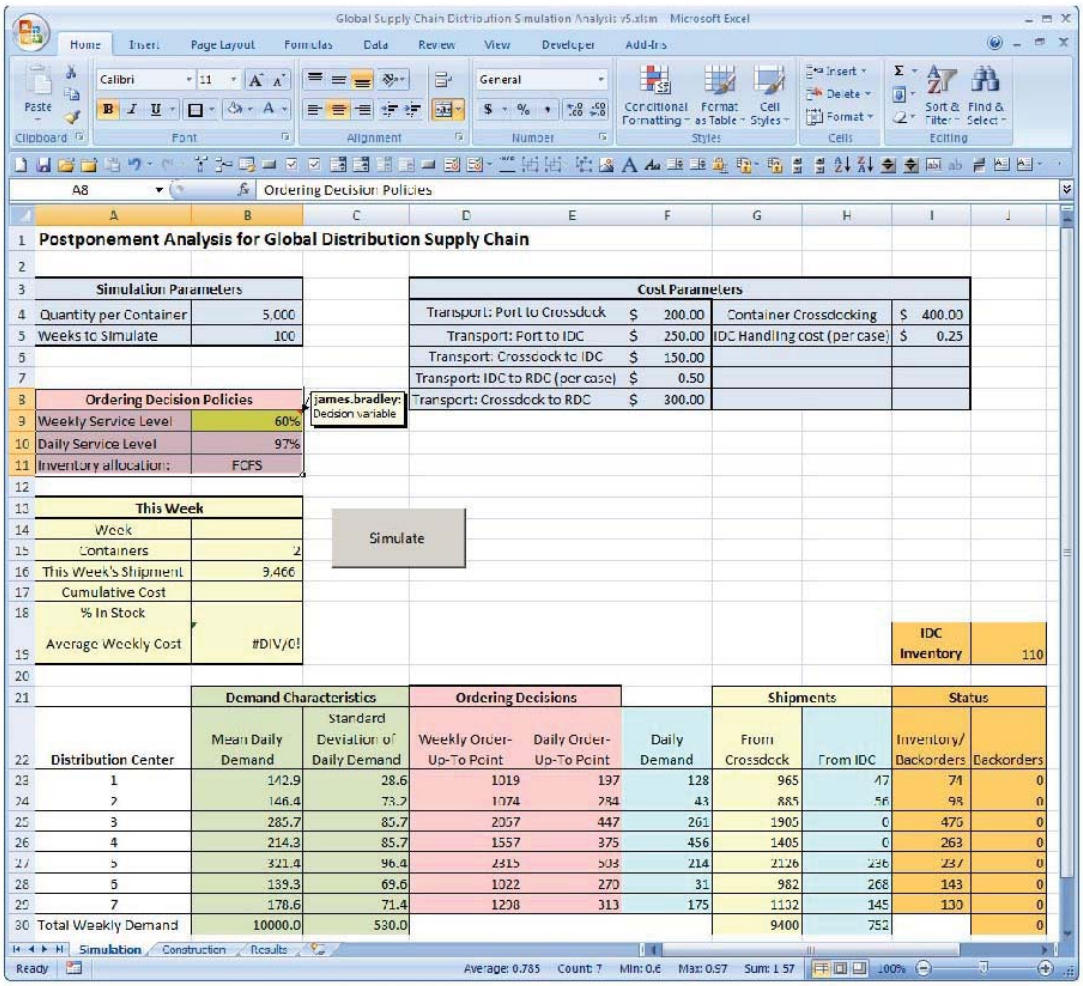

Figure 2: Main Worksheet of Simulation Workbook

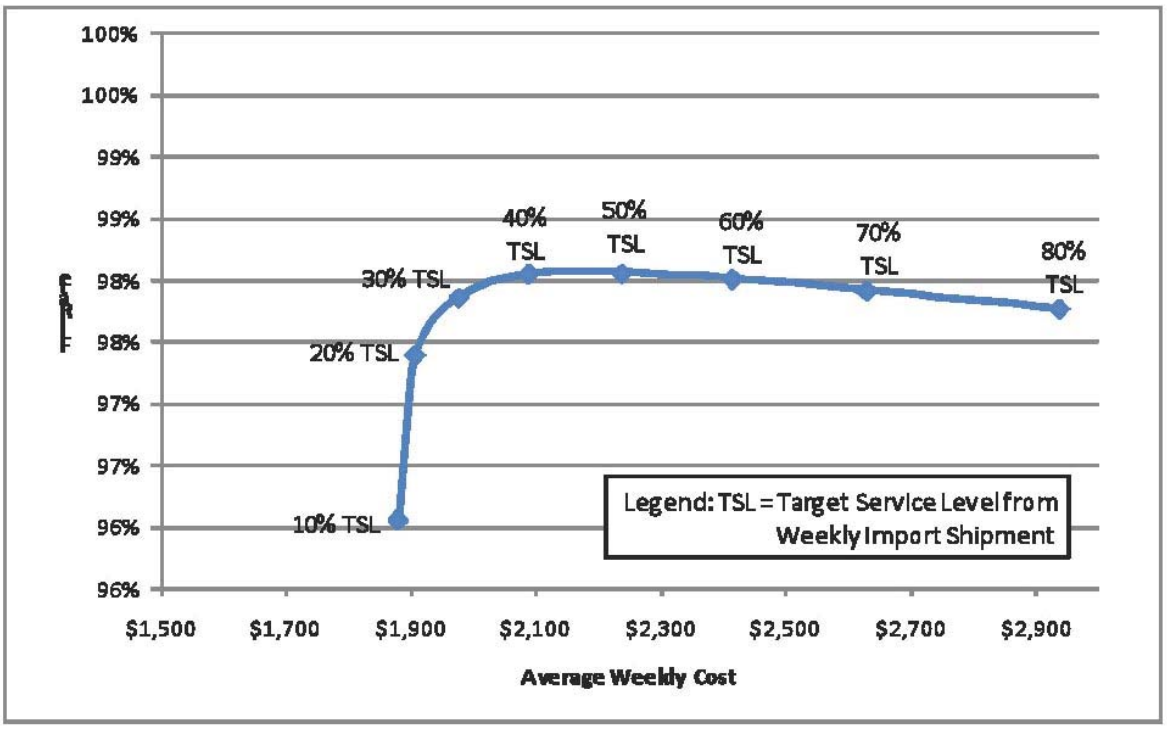

Figure 3: Simulation Results

1. General programming knowledge
    a. Basic understanding of a computer program as a set of steps that are automatically executed
2. Excel spreadsheet knowledge
    a. Understanding of manual vs. automatic recalculation
    b. Named ranges
    c. CommandButton Form Object
3. VBA programming knowledge
    a. Programming statements
        i. Declaring variables
        ii. Basic arithmetic operators
        iii. Worksheet.Range.Offset.Value
        iv. Application.ScreenUpdating
        v. Application.Calculation
        vi. Range.Calculate
        vii. For-Next loops
        viii. If-Then
        ix. With statement
    b. VBA concepts
        i. Difference between changing a cell value versus the value of a VBA variable
        ii. Integer/Double variable types
        iii. Basic Debugging and break points
4. Statistical knowledge
        a. Hypothesis testing
        b. Law of Large Numbers
        c. Central Limit Theorem

The VBA With statement in the list above has not yet been mentioned. While this statement is not necessary to implement the example simulation, it does render the code more readable and benefits the students in later programming ventures. Additionally, we have not discussed basic debugging skills, such as setting break points which, while not absolutely necessary, are useful.

Instruction in VBA programming has been included in several courses at the Mason School of Business at the College of William and Mary. From this experience the author estimates that, in either the undergraduate or MBA programs that the Excel and VBA skills in the list above can be taught in four 1.5 hour class sessions, with follow-on homework assignments. People with less comfort or experience with Excel may require more sessions.

If one had a goal of educating people in industry to do a simulation from which valid conclusions were drawn, we should expect the foundation of many people in industry to be comparable with the skills of a person entering an MBA program. In that case, we should expect the training requirements for such a person to be commensurate with the training required for MBA students, with the exception that some of the statistical training that an MBA student receives may be required. Thus, one could draw the conclusion that a person from industry could be trained with the necessary skills to complete a valid simulation study in a reasonably short time.

4.0 CONCLUSION(S)

This paper documents a discrete-event simulation program which could be used by companies to effectively inform an important supply chain decision. The analysis of the skills required to construct that program, which are also sufficient for other similar simulation models, suggests that simple, yet effective simulation skills can be taught to a many workers in industry. This is but one example of what one might call “Desktop Modeling and Simulation.”

5.0 REFERENCES

Guerrero, Héctor H. (2010a), *Excel Data Analysis: Modeling and Simulation*, Springer, New York.

Guerrero, Héctor H. (2010b). “Essentials for Monte Carlo Simulation with Excel,” submission to MODSIM World 2010 Conference.

6.0 ACKNOWLEDGMENT(S)

The author thanks Héctor H. Guerrero, who has been a wonderful colleague and from whom the author has benefitted enormously in discussions as they have developed the notion of Desktop Modeling and Simulation.