Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

IMPLEMENTING A SIMULATION-BASED SCHEDULING SYSTEM FOR A TWO-PLANT OPERATION

Jeffrey A. Joines
Andrew B. Sutton
Kristin Thoney

College of Textiles
North Carolina State University
Raleigh, NC 27695-8301, U.S.A.

Russell E. King
Thom J. Hodgson

Industrial Engineering
North Carolina State University
Raleigh, NC 27695-7906, U.S.A.

ABSTRACT

Scheduling any complicated job shop becomes increasingly more difficult when the cycle time is reduced. This paper will discuss the implementation of a simulation-based scheduling system that properly schedules parts in a two-plant operation. The system has allowed the company to reduce the cycle time by at least a week from two/three weeks to one/two weeks. As part of the project, the generation of the input data needed to drive the simulation is also discussed since this data did not exist in the correct form. The model generation, simulation development, and experimentation will be discussed. The system that is described is currently being used to generate the schedules.

1 INTRODUCTION

Recently, opportunities for cycle time reduction, which exist across an organization, have received much attention. The potential for improvement appears to be greater across the entire supply chain environment rather than on individual entities within an organization. These entities could be different plants, different functional areas with in a job-shop, etc. Most of the scheduling analysis has been done at the macro level and little research has been conducted on the impact of coordinated, detailed production scheduling between different entities in the supply chain and transportation.

A great deal of research has been focused on solving the job-shop problem, over the last forty years, resulting in a wide variety of approaches. Recently, much effort has been concentrated on hybrid methods to solve them as a single technique cannot solve this stubborn problem. As a result much effort has recently been concentrated on techniques that combine myopic problem specific methods and a meta-strategy, which guides the search out of local optima. These approaches include Tabu Search, Hybrid Genetic Algorithms, etc. However, these methods still suffer from some of the same problems. They can not solve industry-sized problems with tens of thousands of part operations, with hundreds of machine types each having multiple machines of each type, with workers being a constraint (i.e., more machines than workers), the current state of the system (i.e., WIP), etc.

Therefore, simulation-based scheduling systems offer the ability to accurately model the current system. Many of the current major simulation vendors offer scheduling modules. The goal of any system is to develop schedules very quickly such that decisions can be made and alternatives tried. The Virtual Factory (VF), developed at NC State is one such tool that has been found to provide near-optimal solutions to industrial-sized problems in seconds (Hodgson *et al*. 1998). The VF is a simulation-based procedure, which solves deterministic problems by minimizing the maximum lateness, $L_{\max}$.

This paper will discuss the implementation of a VF derived scheduling methodology for a garment maker that has its operations in a two-plant facility. The facility discussed is a cutting operation that takes fabric and cuts them into garment pieces and then ships sew kits out to various places in South America and the Caribbean to be sewn together to make the garment. The two plants are approximately 30 minutes apart by truck.

In Section 2, the description of the job shop is described in detail. What had to be done in order to obtain the data to drive the detailed system is presented in Section 3. Section 4 and 5 will describe the scheduling system that has been developed and implemented, which is derived from the commercial version of the VF. Section 8 presents the conclusions and future research.

2 PROCESSING SUMMARY

The facility has only five major processes (see Figure 1), which may seem to be easy but the individual processes are very complex. The processing begins with the orders

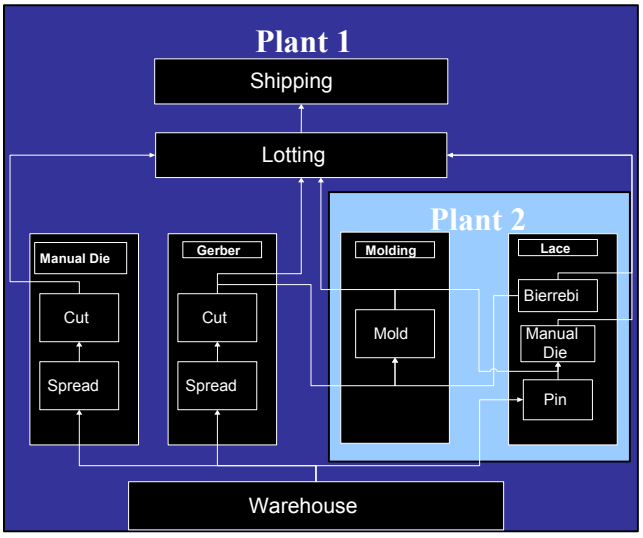

Figure 1: Job Shop Scenario

coming in from the production planning system typically on a weekly basis, every Sunday evening. However, only 80% of the orders will be available while the remaining orders will come in during the week forcing the system to reschedule potentially every day.

Those orders are then exploded into garments and ultimately the parts that make up those garments. During the next process, certain parts are assigned to like material groups. Then marker planning will take these groups assign them into spread groups and develop the markers to be cut. Multiple parts from different orders maybe grouped together on a single spread. This is an effort to increase productivity and material usage in cutting.

Spreading is the first major process, which consists of spreading out the material (rolls of fabric) to allow the parts to be cut out of the material. Also, certain types of fabrics have to utilize specific spreaders. The next process is the actual cutting of the material, which can take place on automatic Gerber cutters or die cutters. Capacitated queues exist between the spreaders and the cutters. The spreading and cutting of the material takes a week in the current situation. Once the parts are cut they can be sent either to Lotting along with lace and die cut parts or to molding. Both the spreading and cutting operations are group operations (i.e., parts are processed together in batch before they can be sent to the next operation).

Some of the parts in certain garments require a molding process, which is actually done at the second plant. The molding operation is one of the most complicated processes owing to the constraints. One, they are tooling restrictions based on the actual molds and the sheer number of molds of a given type and size. The most complex problem is the operation has sequence dependent setup times to switch out the tools. Different sizes of the same type require a slight setup (20 minutes), while different types require a longer setup (40 minutes) and finally changing from a dark color to a light color will require a different setup as well of 10 minutes. Sequence dependent setup capability has to be included in the methodology. Once parts leave the molding operation, they will proceed to either Lotting or some of the parts require another die cutting operation. It takes approximately one week for parts to be molded and returned back to the original plant.

The final major operation (Lotting) occurs in the original plant that takes all of the parts for a garment and assembles them into sew kits. This processing is not started until all of the parts of a kit are available (i.e., no partial Lotting is allowed). Therefore, garments with molding operation take three weeks, while the non-molded garments take only two weeks to process.

In the current situation the garment maker was scheduling locally (i.e., each process was developing a schedule that optimized its process) based on many different criteria from longest processing time, to day of the week, shortest processing time, minimizing setups, etc. Under the current scenario, each process had weekly buckets to develop schedules. For example, the molding supervisor knew what its weekly buckets would be and could easily minimize setups as well as leave jobs that were easy to do on Friday while Lotting always processed the jobs in longest processing time order.

In an effort to cut the overall lead-time to produce the garment to a finished good, management decided to reduce the cycle time to process the garments in these two plants by at least one week. Also, trucks picked up parts based on what they could carry owing to capacity considerations rather than on what is needed at the next operations. The facility had a good track record of shipping the garments out in the current cycle time based on the weekly buckets. However, the facility cannot continue to schedule based on the current local criteria and be able to ship the garments out on time in a tightened cycle time.

This means the facility is going to have to schedule the garment parts more efficiently in a global fashion to be able to meet the reduce cycle time. All of the processes will need to be linked together tightly with an overall global criteria with a common scheduling methodology. The system will need to generate priority lists for each machine in each operation as well as maintain any restrictions on the machines owing to sequence dependent setups, tooling and worker considerations, trucking between plants, grouping of parts into spreads, etc.

3 DATA DEVELOPMENT

Based on the requirement to reduce the cycle time, a detailed scheduling system is required owing to the complexity of the system (i.e., the number of part operations, the sequence dependent setups, the trucking between plants, the number of operators, the number of machines for each type, etc.). A detailed scheduling system will determine for each machine the

priority list of parts to operate on. At the time of the development of the scheduling system, the processing information in the databases was at the garment level (i.e., the database could tell that it took 32 minutes to Gerber cut all of the parts in the garment but not the processing times for the individual parts). This level of processing times is not sufficient in determining a detailed schedule. A methodology had to be developed to generate the data needed to be able to produce a viable simulation. As will be seen in Section 5, the data generation methodology is independent from the actual scheduling system (i.e., the data is generated outside the system and then passed to the system).

What has been the case for many other implementations; the majority of the time was spent cleaning up the data before we could even think about generating the required data. Part of the reason, the data has never been used to drive anything operational like a simulation or a scheduling system. For example, products in the database that had been discontinued needed to be removed and the individual parts in the database had to be matched up to the proper products in order to link to garment time standards. Also, because several different systems had been combined over the years, inconsistencies in data had to be cleaned up (e.g., 2XL versus XL2). Other data issues were encountered but are proprietary to their systems and cannot be discussed.

Once these data issues were resolved, the garment level time standards could be manipulated into part time standards. Owing to the number of parts and the timeframe, it was not practical to pursue a time study. However, if more accurate methods of generating the part time standards are determined, the scheduling methodology that has been developed will only get better.

The processing times for the individual parts of the garments would be generated from the processing times of the garment since the total processing time of each garment at each process is known. Also, the material usage for each part of the garment is known. With the assumption that the area of the garment is a direct correlation of the processing time (e.g., the cutting time for a part would be directly related to the amount fabric that is being cut out) the processing time at each operation can be determined based on a ratio at the part level. The following formula was used to determine an individual parts material usage:

$$\%\,Material\,Usage = \frac{Material\,Usage}{Total\,Material\,Usage} \tag{1}$$

Now using the part's percent material usage, the processing time of the part on an operation can be determined using Equation (2).

$$\begin{aligned}&\text{Part Processing Time} = \\ &\text{Garment Processing Time} * \%\,\text{Material Usage}\end{aligned} \tag{2}$$

The appropriate queries and data tables were generated using these equations. Now the download of the current set of orders and ultimately part operations with processing times could be generated consistently every time. This was a major accomplishment in having the proper data needed to input into the simulation.

4 VIRTUAL FACTORY

Now, the scheduling methodology needs to be designed. Hodgson et al. (1998, 2000) developed a job shop-scheduling algorithm and named it the Virtual Factory (VF). The VF is an iterative, simulation-based procedure, whose objective is minimizing maximum lateness. It has been found to provide near-optimal solutions to industrial-sized problems in seconds. Thoney et al. (2002a) expanded the VF to include inter-factory transportation operations, which enabled the detailed scheduling of entire multi-factory manufacturing supply chains. Although performance was found to be good when transportation was not a bottleneck, the scenarios were tested in a transient setting. Starting and ending effects were observed to impact performance. The more realistic rolling horizon setting explained in Thoney et al. (2002b) would enable us to more accurately test how the VF would perform in multi-factory settings in industry by helping to eliminate transient effects. Using the rolling horizon algorithm, a variety of experiments were undertaken to gauge performance under different conditions.

4.1 Slack Criteria

Slack Value is defined as the difference in time that the job will take to process and the amount of time allocated to it or the measure can be defined as the amount of delay that can be put on the part and still be finished on time and can be used to prioritize the jobs in the simulation. Let d_i be the due date of job i and p_{ij} be the processing time of job i on machine j. The slack of job i on machine m is computed as

$$Slack_{im} = d_i - \sum_{j \in m^+} p_{ij} \tag{3}$$

where m^+ is the set of all operation on job i's route subsequent to the one performed on machine m. This formula is often used in most scheduling algorithms and is a measure of the amount of time a job can queue and still meet its deadline (Hodgson et al. 2000). However, it has been shown that slack does not perform well as a dispatching rule in early experiments found in the scheduling literature.

To remedy this situation, a revised slack value that incorporates queuing times is used as the sequencing rule in

the Virtual Factory. The revised slack for job *i* on machine *m* is computed in Equation 4:

$$Slac\,k'_{i,m} = d_i - \sum_{j \in m+} p_{ij} - \sum_{j \in m++} q_{ij} \qquad (4)$$

where m++ is the set of all subsequent operations to machine *m* on the routing sheet for job *i,* except the immediate subsequent operation. The simulation is run until the lower bound is achieved or a specified number of iterations is reached, and the best solution is saved. Determining and estimating the queuing time of the subsequent operations is the difficulty in using revised slack. One way to determine queuing time is to use historical data. However, this method is very inaccurate since it does not take into account the current load and status of the system. Another way is to use a simulation-based scheduling system like VF to simulate the system under current conditions estimating the queuing times. Queuing times are recorded for each job at each machine it visits in one iteration of the simulation and used in the next iteration. This process is repeated until it converges and has been shown to produce optimal or near optimal results on a certain set of problems (i.e., ones without sequence dependent setup of times like the molding operation). The idea of slack is to force jobs, which have very little slack to become priority jobs to be handled first in order to force these jobs, or parts to be processed next.

Even though revised slack has enhanced slack, some parts (i.e., molded parts) have setup times that have to be taken into account. Set-up times within the molding department can be added by subtracting out the setup time of job *i* on machine *m* for a modified revised slack:

$$Slac\,k''_{i,m} = d_i - \sum_{j \in m+} p_{ij} - \sum_{j \in m+} s_{ij} - \sum_{j \in m++} q_{ij}\,. \qquad (5)$$

Since slack is calculated when a part is put into queue, sequence dependent setups cannot be taken into account using this method. Essentially, if you always have non-sequence dependent setups, then the setup just increases the processing time by the amount of setup.

The overall global goal is to produce the parts on time to be shipped while the local goal is to have as few setups as possible. An iterative algorithm that works like the following has been employed. Revised slack ($Slack''_{i,m}$) will be used to prioritize parts at each machine center using a specified setup time for all sequence dependent operations (e.g., 30 minutes for all parts in the molding operation). This number was chosen since it represents the average of the setup times for the molding operations but not the worst case. Once the algorithm has converged to the best solution and then using a specified time window amount (*tw*), reorder the priority list within this window to minimize setups. Repeat for all subsequent time windows until all parts have been sorted. This represents a myopic view but will minimize some of the setups while maintaining the overall goal of due date shipment.

5 MODEL LOGIC

To be able to develop the system, the scheduling logic must be developed in order to correctly describe the algorithm to schedule the parts. The logic seen in its entirety in Figure 2 and in viewable sections in Figures 3-9 was developed during the summer by a set of interns.

The first group of blocks (Figure 3) imports the current set of remaining and new parts into production data-

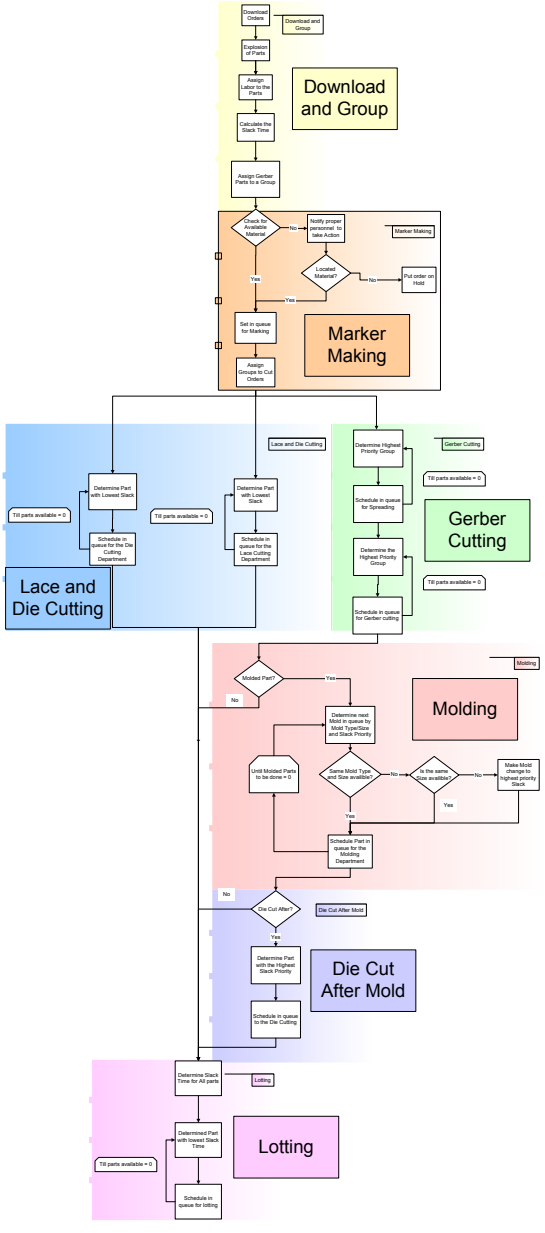

Figure 2: The Scheduling Logic

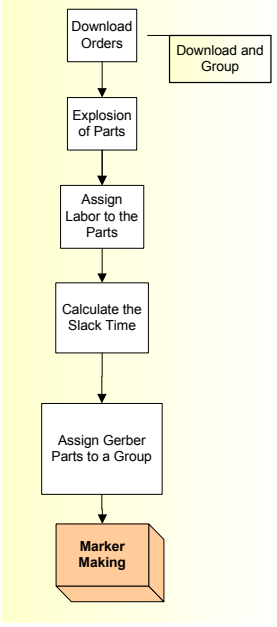

Figure 3: Download and Group Parts into Marking

bases. The orders are "exploded" into all the parts required to fill the orders. The labor is also assigned to the part operations from the labor database and then slack time is calculated for the orders. In order to maximize material usage parts are assigned to like material for cutting.

Figure 4 details the next section, which checks for the availability of the materials. If the material is not available, flags are sent to determine when it will be available. If the material is available, then the order is set in queue based on the slack time. The groups are then broken down to smaller cut orders for processing purposes.

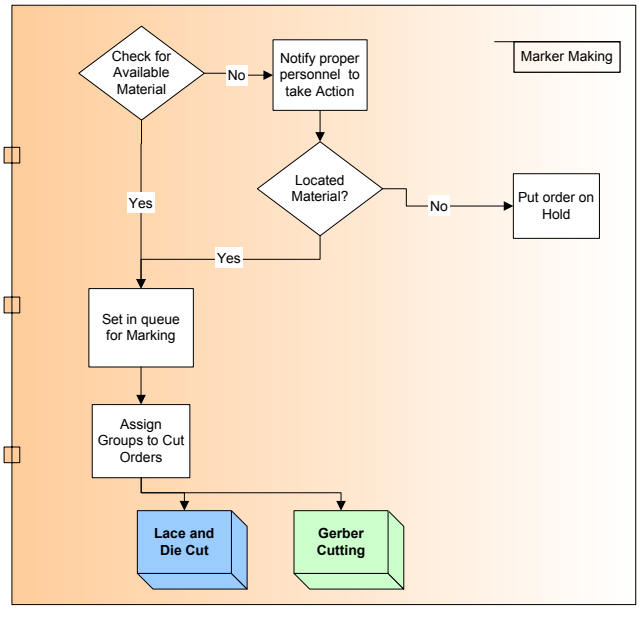

Figure 4: Marking Making for Cutting

The cut orders then are divided into three routes, depending on their process, such as Die Cutting, Lace, and Gerber cutting. Die and Lace are basically the same (see Figure 5) schedule the individual parts based on revised slack. The Gerber process has additional spreading processes as seen in Figure 6. The only difference is these two operations are scheduled based on the material group, not the individual parts. The highest priority group (minimum slack value for all parts in that group) is determined before each process (Spreading, and Gerber Cutting) and then the grouped part is placed in queue.

All non-molded parts proceed to Lotting along with the lace and die parts. The molded parts proceed to the molding logic (see Figure 7), which is very complex. The complication rises due to the constraint on molds type and

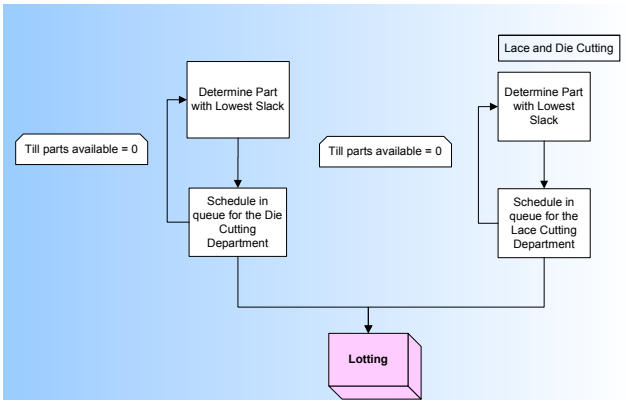

Figure 5: Lace Cutting and Die Cutting

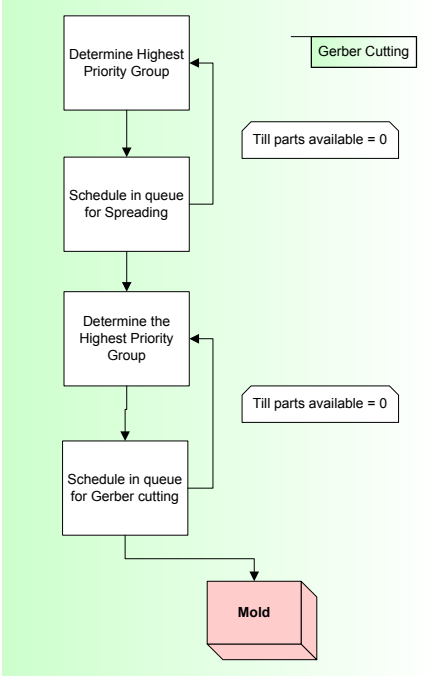

Figure 6: Gerber Cutting Logic

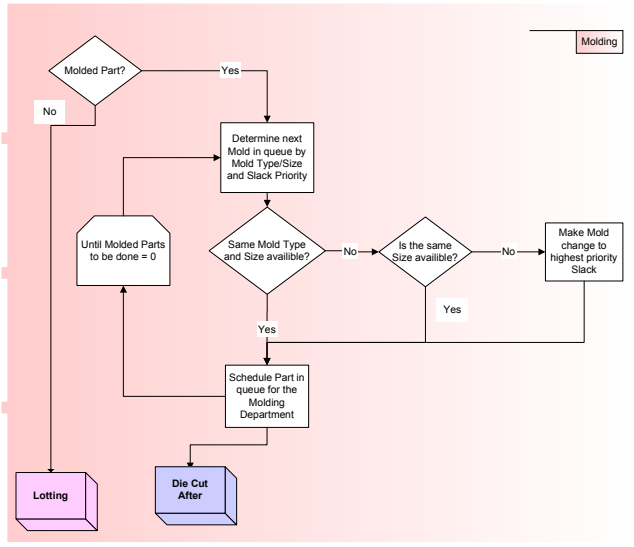

Figure 7: Molding Department Cutting

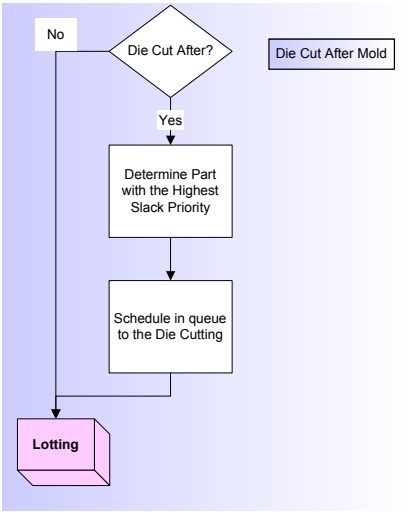

Figure 8: Die Cut after Molding Logic

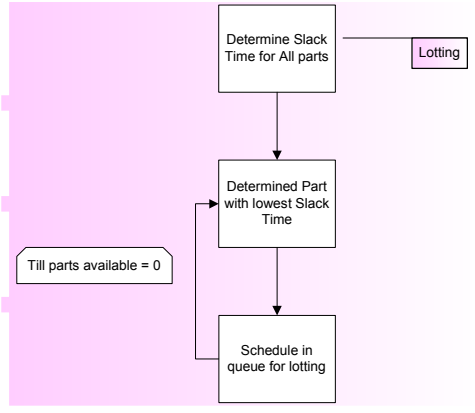

Figure 9: Lotting of Parts into Sew Kits

size and sequence dependent setup times. Changing a machine's mold type/size is time consuming and every effort is made to minimize setups. The next part to operate on is determined based on the slack value. If the same mold type and size is available then the part will follow next one in the queue on that particular machine. If a machine with the same size is available, then those parts will be placed in its queue since the changeover time for the same size mold is shorter than a complete setup. If this does not exist, then a complete changeover will be made for the parts with the highest priority slack and be operated by the next machine available for changeover.

Next, some molded parts are die cut after the molding process (see Figure 8). If die cutting is not needed they proceed to Lotting otherwise those parts are placed into queue for processing based on revised slack.

The final process (see Figure 9) is the Lotting process where all the parts of a garment are assembled into sew kits. Again, revised slack is used but it is based on the main assemblies (sew kits) and is only determined if all parts for the kit have arrived at the department since lotting partial kits is against policy. The highest priority main assembly is determined and scheduled to be lotted.

6 MODEL DEVELOPMENT

The previous section developed the logic that needed to be performed in order to schedule the two plants and the five major areas (Gerber Cutting, Die-Cutting, Lace, Molding, and Lotting). To determine the revised slack calculations with setups which ultimately will generate the priority lists at each machine, the commercial version of the Virtual Factory was used to simulate the system. The VF is an object-oriented simulation written in C++ and linked to a Visual Basic.Net™ interface with MS Access™ as the underlying database engine. The goal of the VF implementation is for the engine (simulation-based scheduling system) to be independent of any scenario (i.e., data would drive the simulation). The interface can be easily tailored to fit the current scenario that is being scheduled. The only thing that is specific in the engine is in the read in function, which reads in the resource groups, machine centers, and parts and their operations from the database to create the simulation.

In the original VF, the system consists of several different objects. *Resource Groups* hold a series of related machine centers where a *Machine Center* is a type of machine. Each *Machine Center* contains a number of machines that can be used. *People*/workers are assigned at the resource group level since it assumed that workers can work any machine within the *Resource Group* because in many cases the number of machines is greater than the number of workers. For example, there is a separate *Resource Group* for Gerber Spreaders and Gerber Cutters because the workers are not cross-trained. Next, the engine contains *Orders* that are made up of *Main Assemblies* (e.g., in the garment these represent the sew kits). The main assemblies are made up of all of the *Parts* that are contained

in this upper level where a part can be a individual part or sub-assembly that require subsequent operations. The *Part* contains the routing information, which in turn contains all of the operations along with processing times, setups, etc.

The VF simulation engine is very generic in the since that parts basically flow into a resource group and get processed by the machine centers in that group. The parts will move from one resource group to another until they have finished with all operations. Once they have finished, they release their parents (sub-assemblies) or the main assembly. Once all of the parts of a sub-assembly have finished it is released into the system. When all of the parts of the main assembly have done, the order is updated. The simulation continues until all orders have been fulfilled.

6.1 Object-Oriented Simulation

However, many changes had to be made in order for the current version of the VF to fit the garment scenario. The current system does not handle sequence dependent setups, parts that are grouped together to be processed and then released into the system (i.e., disassemblies). Since the engine is based on an Object-oriented simulation, adding additional functionality is quite easy.

Most simulation languages are object-based (i.e., a resource object, queue object, etc.) The object-based approach only allows extensibility in the form of composition (i.e., new objects can only be created out of existing objects). An object-oriented simulation deals directly with the limitation of extensibility by permitting full data abstraction. Data abstraction means that new data types with their own behavior can be added arbitrarily to the programming language. When a new data type is added, it can assume just as important a role as any implicit data types and can extend existing types (Joines and Roberts, 1997, 1998). For example, a new user-defined machine class (e.g, molding) can be added to the language that contains standard machines without compromising any aspect of the existing simulation language, and the molding machine may be used as a more complex machine. There are two basic mechanisms in C++ that allow OOS to provide for extensibility: **inheritance** and **genericity**.

6.1.1 Inheritance

Inheritance allows classes to exploit similarity through specialization of parent classes (i.e., child classes inherit the properties of the parent and extend them) as seen in Figures 10 and 11. All *Machine centers* have an associated list of machines, a queue store parts and the appropriate data methods to specify these properties. The virtual methods (bolded functions) represent key functions that more specialized classes can override. The object stores a set of machine centers. If the *Resource Group* tells the machine center to seize a machine (i.e., the **seizeMachine** method is invoked), the simulation at runtime will determine if it is a

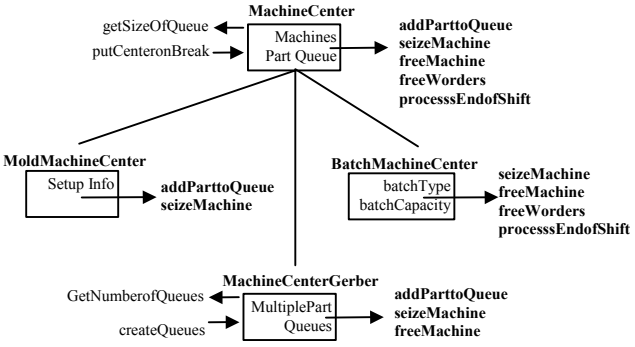

Figure 10: Partial Machine Center Class Hierarchy

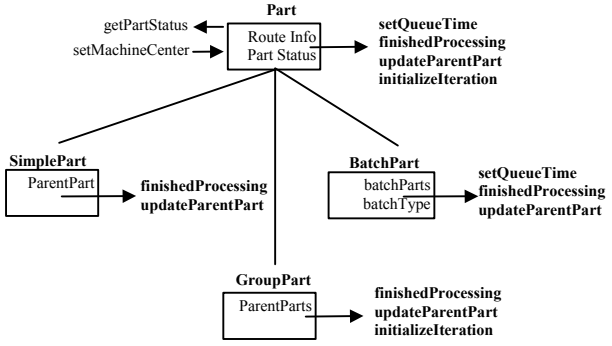

Figure 11: Partial Part Class Hierarchy

batch machine, simple machine, or molding machine. Therefore, specific machine center types can be added without modifying the underlying simulation-based scheduling engine (see Figure 10). Notice the *Batch Machine* inherits all of the properties but adds properties batchType and batchCapacity and redefines several key functions like **seizeMachine** to add the correct func-tionality**.** Now, when a machine center of type batch is told to seize a machine resource, it will now work a number of parts at once based on the capacity of the batch machine. Trucking is considered a batch machine type of process.

Figure 11 shows a partial listing of the *Part* inheritance hierarchy where the *Part* is the key object in the engine since it is moved through out the system of machines, machine centers, resource groups, and plants. It provides properties of the routing information (i.e., operations) and part status plus key virtual methods. *Simple parts* provide the ability of sub assemblies. A part can have many different children if it is a sub-assembly but it could have only one parent indicating either a sub or main assembly. *Simple parts* are created at the creation of simulation while *Batch Parts* are created dynamically when needed and they contain all of the parts in that batch. For this garment, the ability to have dissembles was needed in the spreading and cutting operations. Therefore a new type of part was created and easily added to the system. Group parts are similar to batch parts except they are known at the start of the simulation and slack is calculated on these type of parts and multiple parent parts.

The ability to add functionality with out breaking the underlying engine minimizes the development time. However, the underlying object-oriented design has to be carefully designed. The simulation scheduling engine was developed from the object classes in YANSL. YANSL is an acronym for "Yet Another Network Simulation Language" and is just one *instance* of the kind of simulation capability that can be developed within an OOS environment. For more information on object-oriented simulation, see Joines and Roberts (1997 and 1998).

6.2 Simulation Algorithm

Once the OOS had been modified the following algorithm is used to actually schedule the plants.

1. Download the current status of the plant including number of machines, number of workers, new parts and the remaining operations of the parts already in process.
2. Dynamically create the simulation using the current download by creating machines, resource groups, parts, events, etc.
3. Initialize the simulation to the starting condition (i.e., all machines are idle, workers have been removed, and parts are at the proper machines with their routings set back to the beginning.
4. For the first iteration, simulate the system until all parts have been finished by ordering the queues at the various machines based on slack while recording the queuing time that occurs for each part at each operation as well as the maximum lateness for all parts.
5. Repeat step three and four for a specified number of iterations. However, for the subsequent iterations, order the parts using the revised slack calculation and the queuing time recorded in the previous iteration.
6. Using the queuing times from the iteration which had the minimum maximum lateness, rerun the simulation recording statistics (i.e., priority lists of every machine, the lateness of the individual of the orders, etc.)
7. Since the current VF algorithm does not take into account sequence dependent setups, the sliding window method of minimizing setups is done post simulation. Using the priority lists that are generated for the molding machines, reorder the lists in *n* hour time block to minimize the setup time that occurs where *n* is the specified window block (e.g., eight hours).

6.3 Interface

Once the engine was designed and working, the visual interface needed to operate the engine was built. The interface is tailored to the needs of the garment maker and was written in Visual Basic.Net™ interface with the underlying database of MS Access™. The data needed to drive the engine can be divided into two major groups: static/semi-static and dynamic data. The interface maintains and manages all of the static data like the machine center types, the number of machines of each type, resource/functional groups and the semi-static data like shift information (number of workers, beginning and ending times, etc.) and tooling information. This type of data is not changed very often and therefore is maintained locally. We consider the download of parts, part operations, and orders to be dynamic since it changes daily and is not maintained locally. Instead, it is imported into the local database from a separate data source. Therefore, the engine is independent of the data source. If the data structure is changed, then the import facility is modified to accommodate the changes but the engine remains intact.

The interface provides the ability to modify the static data, import the dynamic data, run the scheduler, and view output statistics, priority lists, critical graphs from the output of the scheduler. The interface has the ability to allow the company to modify and create new output reports that the system can use. Figures 12 and 13 show an example of a priority list for a Lotting station.

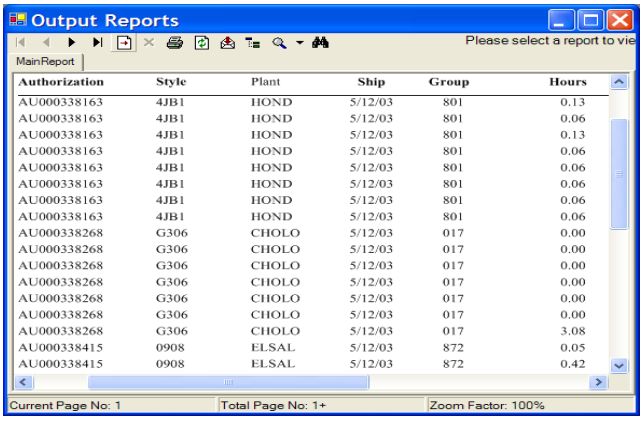

Figure 12: An Example Priority List

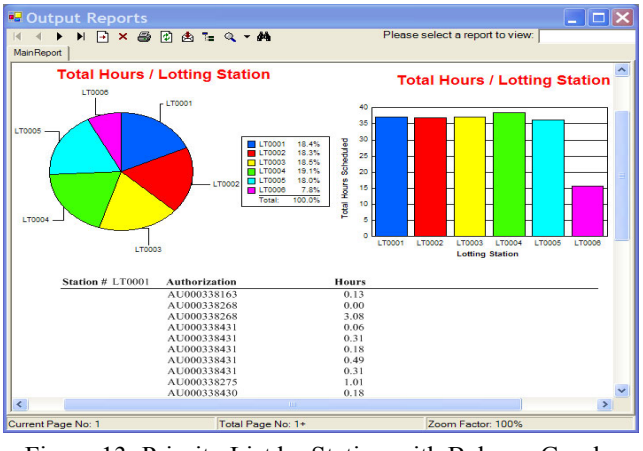

Figure 13: Priority List by Station with Balance Graphs

7 CONCLUSIONS

The simulation-based scheduling system has been implemented and is currently in use by this garment maker. Lessons that were learned that accurate data is a huge issue owing to the fact that the data has never been used to drive something detailed and operational. As the key processes and the data were described, certain assumptions were made in the model that turned out to be wrong. Most of these dealt with the data download. The actual interface was more critical to the company than the quality of the schedules. Appropriate types of graphs, reports, and data management had to be put into place before acceptance.

Second, there has to be "buy in" from all levels (management, supervisors, workers, etc). After the engine began generating actual schedules, many people had difficulty believing the schedules and did not follow the priority/dispatch lists. They spent time modifying the schedules. Lucky for the project, upper level management believed in following some list whether or not it was these lists. Therefore, they convinced everyone that they should follow the priority lists exactly until something went wrong which would allow us to fix anything. After a couple of weeks, confidence in the schedules grew and the people have become dependent on the schedules generated and when they are not they complain.

REFERENCES

Hodgson, T.J., D. Cormier, A.J. Weintraub, and A. Zozom. 1998. Satisfying due-dates in large job shop. *Management Science* 44 (10): 1442-1446.

Hodgson, T.J., R.E. King , K.A. Thoney, N. Stanislaw, A.J. Weintraub, and A. Zozom. 2000. On satisfying due-dates in large job shop: idle time insertion. *IIE Transactions* 32: 177-180.

Joines, J.A. and S. D. Roberts. 1997. An Introduction to Object-Oriented Simulation in C++. In *Proceedings of the 1997 Winter Simulation Conference*, ed., Sigrun Andradottir, Kevin J. Healy, David H. Withers, Barry L. Nelson, 78-89. Institute of Electrical and Electronics Engineers,

Joines, J. A. and S. D. Roberts. 1998. Object-oriented simulation. In *Handbook of Simulation*, ed. J. Banks, 397-428. New York: John Wiley & Sons.

Thoney, K.A., T.J. Hodgson, R.E. King, M.R. Taner, and A.D. Wilson. 2002a. Satisfying due-dates in large multi-factory supply chains. *IIE Transactions* 34: 803-811.

Thoney, K.A., J.A. Joines, P. Manninagarajan, and T.J. Hodgson. 2002b. Rolling Horizon scheduling in large job shops. *Proceedings of the 2002 Winter Simulation Conference*, ed. E Yucesan, C.-H. Chen, J.C. Snowdon, and J.M. Charnes, 1891-1896. Piscataway, New Jersey: IEEE

AUTHOR BIOGRAPHIES

JEFFREY A. JOINES is Assistant Professor of Textile Engineering at NC State University. He received his B.S.I.E., B.S.E.E., M.S.I.E, and Ph.D. from NC State University. He received the 1997 Pritsker Doctoral Dissertation Award from IIE. His research interests include evolutionary optimization, object-oriented simulation, simulation-based scheduling and supply chain optimization. His email and web addresses are <JeffJoines@ncsu.edu> and <http://www.te.ncsu.edu/joines>.

ANDREW B. SUTTON is a graduate student in Textile Engineering at NC State University. He received his Bachelor's Degree in Textile Engineering from NC State University and his areas of interest include simulation, six sigma and simulation-based scheduling.

KRISTIN A. THONEY, an Assistant Professor in the Textile and Apparel, Technology and Management Dept. at NC State, joined the faculty in 2000. She earned her Ph.D. in Industrial Engineering and Operations Research and her MS in Operations Research also from NC State. Kristin's research interests include production scheduling, logistics, and supply chain modeling. She is a member of INFORMS and IIE. Her email and web addresses are <Kristin_thoney@ncsu.edu> and <http://www.tx.ncsu.edu/faculty_center/directory/>.

RUSSELL E. KING is a Professor of Industrial Engineering at North Carolina State University and has been on the faculty since 1985. Previously, he worked as a Systems Analyst for Dynamac Corporation of Rockville, Maryland and the Naval Aviation Depot at the Jacksonville Naval Air Station. He received a BSSE, Masters, and Ph.D. in IE from the University of Florida. His research interests include control and scheduling of production systems and demand activated manufacturing/supply chains. His email and web addresses <king@eos.ncsu.edu> and <http://www.ie.ncsu.edu/king/>.

THOM J. HODGSON, the James T. Ryan Professor of Industrial Engineering and Director of the Integrated Manufacturing Systems Engineering Institute at NC State. He earned a BSE in science engineering, a MBA in quantitative methods and a Ph.D. in industrial engineering all from University of Michigan. Thom's research interests include production scheduling, inventory control, logistics, real-time control of systems and applied operations research. He is a member of INFORMS and IIE. His email and web addresses are <hodgson@eos.ncsu.edu> and <http://www.ie.ncsu.edu/hodgson/>.