A Scheduling and Capable-to-Promise Application for Swift & Company

Ann Bixby, Brian Downs
Aspen Technology, Inc., 2500 CityWest Boulevard, Suite 1500, Houston, Texas 77042
{ann.bixby@aspentech.com, brian.downs@aspentech.com}

Mike Self
Swift & Company, 1770 Promontory Circle, Greeley, Colorado 80634, mself@swiftbrands.com

Swift & Company uses an integrated system of 45 linear-programming models based on three model formulations to dynamically schedule its beef-fabrication operations at five plants in real time as it receives orders. This scheduling application resulted in documented improvements in key metrics, such as order fulfillment, on-time delivery, and the percentage of a week’s scheduled production for which there are existing orders. The application enabled Swift & Company to execute its business strategy and obtain a 200 percent return on investment in its first year of production.

Key words: industries: agriculture, food; programming: linear, applications.

Swift & Company is a diversified protein-processing business based in Greeley, Colorado. A privately held company with publicly traded debt, Swift & Company has three business segments: Swift Beef, Swift Pork, and Swift Australia. With annual sales of over $8 billion, beef and related products are by far the largest portion of Swift & Company’s business.

Swift has slaughter and processing operations (the latter are also known as fabrication operations) at five plants located in Colorado, Texas, Nebraska, and Idaho. Three of the plants process fat cattle, animals up to two years old that have been fattened on feed lots for 90 to 120 days before slaughter. Lean cattle are older animals, such as bulls or dairy cows, that are not fattened before slaughter. Used primarily in commercial food products, these animals are processed at the remaining two plants. Each plant has the capacity to process approximately 2,500 head of cattle per shift for a total of 18,000 to 25,000 per plant per week. This translates to over 6.0 billion pounds of beef delivered annually.

Swift’s most important product line is boxed beef. Cattle buyers procure cattle on the open market based on projected demand by brand, grade, and weight. After slaughter, each head of cattle yields two sides of beef. After spending 80 hours in a chill room, each side is graded by the USDA for quality and yield. The fabrication process begins as workers divide each side into seven primal cuts: the chuck, the rib, the loins (short loin and sirloin), the round, the brisket, the plate or navel, and the flank (Figure 1). Then, they divide each primal into smaller, subprimal pieces, pack them in plastic, vacuum-sealed packages, and box them for sale. Because beef products are highly perishable, and because of the economics of the beef business, once fabrication begins, workers must completely process all seven primal cuts from each side during the same shift. The work is intense manual labor done by workers with little more than a meat hook and a sharp knife. It takes a great deal of skill to make the proper cuts while minimizing trim waste and maintaining the speed of the production line.

For each primal, there are many possible production pathways, which together form a tree structure (Figure 2). Types of cattle are classified by quality grade, yield grade, weight, and brand characteristics. USDA inspectors determine quality grade (for example, prime, choice, or select) and yield grade. They give higher quality grades to carcasses that show high levels of fat marbling based upon a cut made at the 13th rib. They base yield grades upon the thickness
of the outer fat covering; the greater the fat covering, the lower the percentage yield of lean meat. Like any agricultural product, individual cattle are highly variable and may not grade out as expected. This unpredictability creates variability in the fabrication process. Also, the cattle vary greatly in weight, with a typical carcass weighing between 600 and 800 pounds. Within that range, animals are classed as light, medium, and heavy. Because customers order by weight and not by the piece, it takes a greater number of pieces from light carcasses to fill an order than from heavy carcasses.

Finally, Swift & Company has many product brands. To be sold under a particular brand name, an animal must meet certain standards and possess certain characteristics. For example, to be sold as “certified Angus beef” (CAB), an animal must have a minimum percentage of Angus bloodstream and have a quality grade of at least choice. An additional complication is that some downward substitution is possible although not desirable because such downgrades usually mean loss of margin. For example, a plant could use a CAB primal to fill an order for a commodity product, but that would be an inefficient use of a premium quality carcass.

All of the possible combinations of brand, grade, weight, and yield characteristics add up to hundreds of possible cattle types. Each cattle type has a disaggregation tree for each of its seven primal cuts, with some trees having up to nine levels. Thousands of possible fabrication operations must be considered.

In addition to boxed beef, Swift’s other product lines include offal and ground beef. Offal means the internal organs of the animal, and offal products go to a various markets around the world, typically frozen. The fabrication and disaggregation for offal are similar to those for boxed beef, although less complex. Trim is a byproduct of fabricating boxed beef, and it is characterized by the percentage of lean and fat content. Swift either sells it as trim or uses it to make ground beef. To make ground beef, the plant blends and grinds the trim to a target lean point and coarse-
ness. It then packages it for retail sales or for further grinding at or near the point of retail sale.

Because beef is highly perishable, Swift & Company’s customers specify a maximum age upon delivery, typically 10 to 14 days. Thus, production schedulers must be aware of inventory quantities and of inventory age. The result is an industry with high volume, rapid turnover of inventory, and a brutal velocity. With over 1,500 finished product codes (stockkeeping units (SKUs)) and over 30,000 ship-to locations, Swift has complex opportunities to differentiate within a highly variable manufacturing process. Yet, beef remains a commodity product with prices driven by a commodity market, and despite the complexity in its production, the margins for beef are thin, perhaps one percent or less. The combination of tight margins and a variable manufacturing process have historically led to variability in Swift & Company’s profit margin.

Raw material costs in the beef industry can reach 85 percent or more. Thus, the economics dictate that, to be profitable, businesses must sell every part of each animal or use it in some way. In addition to meat products, Swift sells the hides, hooves, and even the blood. Demand is high for some products, such as the tenderloin and the ribeye, and the competitive market for these cuts is reflected in their prices and margins. Other less desirable cuts create less demand, and Swift & Company must price these products to move them without unnecessarily sacrificing profits.

Tight margins mean that optimizing cattle procurement and product mix is essential to the success of the
business, yet the many sources of variability in raw material, and its high costs, make optimizing both very difficult.

Available-to-Promise (ATP) at Swift & Company As It Was

The high velocity of the beef industry is evident in Swift’s sales and marketing operations. Customer service representatives (CSRs) sell beef over the telephone by talking with buyers from their more than 8,000 customers. In real time, CSRs must be able to see the availability of current and future inventory while considering requested delivery dates and maximum product age upon delivery. They may enter new orders and new line items, and modify or delete existing line items. Sales teams must also monitor projected inventories of unsold items so they can proactively sell them and prevent spoilage.

In the past, Swift & Company’s order-management systems, and business processes attached to them, were incapable of handling the volatility of the business. At the start of each week, CSRs would have sold only a small percentage of the week’s production. Based on the existing orders, the system created a weekly production schedule that tried to fill those orders with the available raw material and with the scheduled production hours. The latter typically included two eight-hour shifts for five days a week, although Swift often added extra shifts on the weekends as needed. Because the business requires the fabrication of each entire carcass during a single shift, planners scheduled production of unsold pieces based on a list of “balance items,” the plants’ best guess about what could be sold during the coming week. As CSRs entered orders, the system could modify the schedule, but only in the direction of further disaggregation. The horizon for the schedule was the end of each production week, so the system could not schedule production at the end of the week to fill orders at the beginning of the following week. As a result, the plants were often unable to fill orders even though they had the raw material and capacity.

The situation was exacerbated by the poor visibility the system provided to the planned schedule. Without a clear line of sight to inventory availability, CSRs often entered orders without knowing whether the plants could fill them. Consequently, the plant schedulers manually altered the production schedule to expedite orders during execution. Plants often shipped orders late, requiring CSRs to contact customers to schedule new delivery dates. Often customers failed to order the balance items, so as the end of the week approached, CSRs offered discounts to move the unwanted products. Not only did this practice reduce already tight margins, but it also made accurate forecasting nearly impossible because sales history did not reflect actual customer demand. This misalignment of demand and production was in part caused by the order-management system’s inability to use the full spans of the many disaggregation trees when committing to or changing customer orders. Using Swift’s legacy system, schedulers could not automatically switch to an entirely different branch of the tree even if the capacity and raw material were available to supply ordered items.

Swift & Company’s managers realized that they were trapped with a production-push business model (Figure 3), which did not work for an industry with such volatility and velocity. The company’s customers were well aware of its problems with execution and reliability; many of them did not regard Swift & Company as a supplier of choice. They also understood Swift’s production difficulties and knew that if they waited until late in each week, when the lots at the

![Figure 3: Prior to Project Phoenix, Swift & Company used a production-push model to drive its business. It purchased raw materials before knowing what its customers wanted and produced what the plants wanted to make. This meant that the make-to-stock inventory was often not what its customers wanted, and it had to use discounting to move the perishable products.](image-url)
plants were full of refrigerated no-name trailers full of as yet unsold products, they could demand large discounts. The resulting distorted sales history misled cattle buyers about what kind of livestock to procure on the open market.

Solution Design

Swift & Company recognized that it needed to reengineer its supply chain and that it needed an enabling technology to improve its sales and manufacturing performance. To guide its search for a partner to facilitate the change to a demand-pull business model, it identified four critical requirements for a new system that would provide a solid technical foundation for its business:

1. To provide CSRs with information on product availability in close to real time,
2. To control inventories accurately,
3. To provide the ability to sell unsold production with maximum margins, and
4. To provide Swift the ability to reoptimize the use of raw material to satisfy changing demand, using the full disaggregation-tree structure.

Aspen Technology demonstrated a mathematical-programming-based solution that was effective in solving Swift & Company’s scheduling and order-management problems. It also provided the rapid data exchange and optimization capabilities needed. Once AspenTech proved the concept, Swift & Company agreed to partner with it to design and deploy a scheduling and capable-to-promise (CTP) application that would transform its business. Swift named this initiative Project Phoenix after a fire destroyed a Swift plant in Garden City, Kansas in December 2000.

In March 2001, a project team started work at the Swift & Company campus in Greeley, Colorado. Swift dedicated 10 full-time employees, including subject-matter experts, a database administrator, software developers, and a project manager, while AspenTech provided four operations research consultants, a data-integration specialist, and a project manager. Throughout the project, both organizations called upon other specialists as needed.

The newly formed team devised a project strategy to take advantage of the similarities between plants and processes in the business. Swift & Company’s beef operations are split into two basic groups, fat cattle and lean cattle. The most important are the fat cattle, young cattle fed grain in commercial feedlots for 90 to 120 days to add additional weight before slaughter. Lean cattle are older, grass-fed cattle slaughtered after their other commercial usefulness has been exhausted. Our plan was to develop a single scheduling model for the Greeley, Colorado fat-cattle plant and, from that base, build scheduling and CTP models for all plants. These models would be identical except for the data used to populate them because all the plants use similar fabrication processes. The plan was to bring the fat-cattle boxed-beef operations on line first, and then the offal and ground beef operations. The lean-cattle plants would be the last to come on line.

Because the CSRs use the CTP functionality in real time while they are on the phone with customers, they needed the quickest possible response time for each CTP transaction. Once we began development, we realized that the level of detail required to create shift-level schedules for the fabrication operations resulted in a model too complex to solve in the few seconds required for the CTP component of the application. Consequently, we decided to break down the problem and create separate models for scheduling and CTP functionality.

The scheduling models produce shift-level and daily schedules over a 28-day horizon. The plant schedulers and shift supervisors use these schedules to plan and monitor each day’s production. The CTP models are the foundation of the order-entry system, and CSRs use them in real time to enter orders and to check product availability. Because the CTP models have a 90-day horizon, we aggregated some model dimensions to reduce the model size and improve the solution time. In addition, we added a third type of model to provide, at 15-minute intervals, snapshots of the available unsold planned production from each shift at each plant to alert the sales teams to undesirable inventory build-ups.

These available-to-promise (ATP) models are periodically refreshed with current inventory, demand, and production data from the CTP models. With five plants and three manufacturing processes at each plant, this meant that the final design included 45 optimization models of varying size and complexity.
that had to work synchronously with Swift’s order-management system 24 hours a day, seven days a week.

With development of the LP components of the solution underway, other team members created the communications infrastructure to support them. We distributed the models on a cluster of multiprocessor Windows 2000 servers to ensure the quickest possible solution times. They are supported by a DB2 database, operating in an AS/400 environment, containing all of the structural and dynamic information needed to create the LP models. This database is also the bridge to inventory and logistics systems. The CTP models are linked to Swift & Company’s order-entry system by Aspen’s CTP engine, a Java-based application that routes users’ queries to the correct model(s) and passes the response back when the model has been solved.

The Scheduling Models
At the heart of the application is a family of LP models that produces a shift-level schedule for each plant over a 28-day horizon. Swift uses these models to fix the production schedule for the next shift and to create a projection of short orders. Cattle buyers provide estimates of future cattle availability, which are combined with information about carcasses already on hand to project the primal-cut availability for the boxed-beef and offal processes. These projections, along with current finished goods inventories, committed orders, and a forecast with committed orders subtracted, constitute the inputs to the models.

We broke down the disaggregation trees into three types of production operations. Cutting operations consume one piece of a primal or subprimal cut and yield one or more subprimal cuts and such coproducts as trim and bone. One can think of the input to such an operation as a parent item and the output(s) as the child item(s). Packaging operations consist of wrapping and boxing finished goods products, with different piece counts for different finished product codes. Bridging operations allow downward substitutions according to brand and grade at various levels of the disaggregation process. Fresh inventory may also be frozen (a process that may take several days) either to satisfy demand for frozen products or because the inventory is past a specified age.

We modeled demand at the order line-item level. Because plants must manufacture all seven primals from each side of beef during a shift, they use a forecast with committed orders subtracted to drive fabrication of unsold pieces. Because Swift did not have a reliable forecast when we started the project, we used the previous year’s sales history as a surrogate. Demand pulls production through the model structure, and the model tries to satisfy all demand with the lowest-cost raw materials and the lowest-cost production methods, with orders taking priority over forecast demand.

Certain fabrication operations are particularly difficult and therefore require more time than other operations. To maintain the desired production rate, schedulers must limit the number of such operations to a specified percentage of the total on the fabrication line for a particular primal cut. Side constraints model this requirement by limiting either piece or box production for some products.

While Swift & Company sells most boxed-beef products fresh, some demand exists for frozen products as well. The company stores fresh inventory at the plants and generally turns it over rapidly. However, it stores frozen inventory at many locations. When combined with the possible brand, grade, weight, and yield combinations for each primal, and with the large number of finished product codes, the potential number of inventory balance rows approaches 900,000. Even worse is the requirement that the model keep track of inventory age. As model time periods roll forward, so must the age of the inventory in days. This leads to a proliferation of inventory variables and a model size that can exceed the memory available on a typical server.

To limit the potential model size, we used telescoping time buckets in the scheduling application. We included seven days at the shift level (with two shifts per day), then seven days at the daily level, followed by a partial week period that goes through Sunday night of the appropriate calendar week. The length of this flexible period is determined by the current day of the week, and it is followed by a one-week period, for a total model horizon of 22 to 28 days.

Characterized by their percentages of lean content, lots of trim are ground to a desired coarseness and blended to a target lean point and then packaged in various sized containers, some for retail sale and
others for sale in bulk to other producers or transfer to another Swift & Company plant. In addition, such products as ground chuck must be produced only from the trim of a particular primal cut. Swift’s ground-beef scheduling models contain a natural LP formulation of a multiperiod blending operation with some additional side constraints. Because of equipment limitations, plants can use a maximum of three or four types of trim, depending on the plant, to produce each product. And because of stringent sanitary requirements, plants conduct a time-consuming cleaning and setup process before changing to a different blend. Therefore, plants enforce minimum lot-size requirements. These constraints require the addition of binary and semicontinuous variables to the blending formulation. Because the raw material available for ground beef depends upon the scheduled boxed-beef production, we must solve the boxed-beef model before the ground-beef scheduling model.

We solve the scheduling models in batch mode at the beginning of the refresh process run twice a day before the beginning of each shift. We update the boxed-beef and offal scheduling models with the current cattle availability, order, and inventory data, and we solve them simultaneously. We post the solutions back to the AS/400 database, and the system passes the projected trim availability to the plants’ ground beef. It solves the ground-beef scheduling models and then posts schedules back to the AS/400 database. The system makes all of the schedules available to each plant and starts the refresh cycle for the CTP models.

The Capable-to-Promise Models
The function of the CTP models is to determine whether a plant can ship a requested order-line-item quantity on the requested date and time given the availability of cattle and constraints on the plants’ capacity during the 90-day model horizon. Each time a CSR enters a query, the system sends the transaction information to the appropriate model, which modifies the existing LP to include the new order information, resolves it, and returns an answer to the order-entry system. Driven by an objective function that has a heavy penalty on unsatisfied demand and using the relevant bill-of-materials information, the model may reconfigure the planned production schedule to accommodate the requirements for the new order or it may find that the plants cannot provide some or all of the requested products on the requested date and time. In either case, it reports the available quantity for each order line item to the CSR through the order-entry system.

To obtain acceptable response times, we developed a separate group of models to handle the CTP functionality in the application. By aggregating all of the possible inventory locations into just two for each plant, a fresh location and a frozen location, we created a model that contained all of the inventory quantity data but was smaller than one containing all inventory variables and inventory balance rows. Further, for the ground-beef CTP models, we relaxed the integer and semicontinuous variable restrictions, reducing the solution times. We also used telescoping time periods, similar to those we used in the scheduling models, in the CTP models with the difference that we added weekly periods to create a total model horizon of 90 days.

During each refresh cycle, after we solve the scheduling models, we take the CTP models off line and restart them with the current information on cattle availability, customer orders, and finished-goods inventory. We load the next shift’s production schedule into the model and fix it, because Swift does not allow new orders or changes to existing orders to disrupt execution during the next shift. However, the schedules for subsequent time periods are flexible. CSRs cannot access the order-entry system while the CTP models are off line, which is not a problem during the morning refresh cycle that takes place at 3:00 am, but it is during the second refresh cycle that runs just before noon. Therefore, we need to make the refresh cycle for the CTP models as short as possible.

Once the CTP models find an initial solution, we put the order-entry system back on line and make the models available to accept CTP transactions. The application handles four types of transactions:

1. Product queries to find out whether a certain quantity of a certain product code can be available on a specified date: In this case, the system saves the LP basis before the transaction and returns the model to its original state after providing a response indicating how much of the requested quantity is available.

2. Order submission: In this case, the system modifies the existing LP to include the new order, finds a feasible solution, or reports that the plants cannot provide all or part of the requested products.

3. Order modifications: If the order is to be changed, the system modifies the existing LP, finds a feasible solution, or reports that the plants cannot provide all or part of the requested products.

4. Order cancellation: In this case, the system removes the order from the model and returns the model to its original state.

The CTP models are designed to provide accurate and timely information to CSRs and other order-entry personnel, enabling them to make informed decisions and respond quickly to customer requests.
(2) Product commitments to both see if the plant can make a certain quantity of a certain product code available on a specified date and to commit to whatever quantity is available: In this case, the LP retains the new optimal solution as the starting point for the next transaction.

(3) Deletion of existing order line items by fixing the unfulfilled demand variable for that line item at the demanded quantity, thus effectively fixing fulfilled demand at zero, to avoid removing rows from the LP solution basis.

(4) General edits of existing order line-item quantities, which can be done in either query or commit mode.

Each transaction may include multiple line items, and CSRs may combine different transaction types. The only restriction is that CSRs may not conduct query and commit transactions together because the former preserves the existing solution basis, while the latter creates a new one. Transactions must be limited to a single plant and to a single customer order number. An individual transaction might contain line items from more than one product line (for example, from boxed beef and ground beef), and thus might require solution of more than one LP model.

Aside from providing accurate solutions, the single most important functional requirement of the application is that each CTP transaction be as short as possible. When a CSR on the telephone with a customer clicks the button on the order-entry screen to initiate a CTP transaction, the application must do the following:

1. Format the transaction and send it to the Aspen CTP engine, which then has to
2. Parse the transaction and send the information to the correct model(s) for that product type and that plant so the model(s) can
3. Parse the transaction data;
4. Update the appropriate LP structures, adding columns and rows to the LP model and perhaps saving the LP basis;
5. Reoptimize the model;
6. Extract solution information and build a response to return to the CTP module;
7. If in query mode, restore the LP to its original status;
8. If in commit mode, fix the committed quantities in the LP structures;
9. Return the response to the Aspen CTP engine, which will
10. Translate and return the transaction to the order-entry system.

In analyzing transaction-processing time, we found that about half of the transaction time typically went into data handling for Steps 1, 2, 9, and 10. This meant that our opportunities for reducing response time lay primarily in the model management and execution steps, 3 through 8. Further, the system must handle as many as 20,000 transactions each week, with as many as 50,000 line items. Peak loads can be as many as 300 transactions an hour, with the boxed-beef models having as many as 290,000 variables and 240,000 constraints. The models' performance must be extremely robust. To accomplish this, the team members from AspenTech devised techniques to enhance the application’s performance.

**Separating Starting Inventory from Production**

Decreasing the length and granularity of the time periods (the original objective was a six-month horizon at the shift level) and aggregating inventory locations was effective but was not sufficient to reduce the model size enough to achieve acceptable response times for the boxed-beef CTP models. The number of possible combinations of product codes and ages for inventory in the model was still enormous. We had to consider inventory from production during the time horizon, which could range in age from zero to 56 days, and the ages of products already in inventory. We could reduce the number of possible ages by treating any inventory over 120 days old as exactly 120 days old. We also decided to allocate starting inventory outside of the model. However, we make allocations from inventory only when we cannot fill orders from new production. In other words, the CTP models first solve an LP to satisfy as much demand as possible from production and then use a heuristic to try to fill orders for any remaining line items from the starting inventory. Inventory allocated to an order line item remains allocated to that line item unless the order is cancelled. This solution includes some double counting, because it may add production to the schedule to fill orders that could be filled from inventory but it ties up as little inventory as possible, increasing the probability that the system will accept near-term
orders that can be supplied only from existing inventory. In the associated scheduling models, demand is static, so they assign inventory first, before scheduling production.

Restricting the Generation of Unnecessary Variables and Constraints
We also improved model sizes and response times by restricting the generation of unnecessary variables and constraints. We used several techniques while processing the model data to reduce problem sizes:

1. We preprocess cattle availability data to determine which pieces and finished products the plant can make in each time period and limit variable generation accordingly.
2. We generate variables to track piece production only when there is a production limit for those pieces.
3. We ensure that boxes are frozen during the shift in which they are produced (that is, no freezing operations from inventory).
4. We generate inventory variables only for ages that are possible from production (that is, age 0 on day 1, ages 0 and 1 on day 2, and so on).

Model Robustness
The CTP application at Swift & Company must be robust to be usable. When the model becomes infeasible for some reason, the recovery process requires reinitializing the model, which includes solving an LP from scratch. It can take as long as 20 minutes to restart the model, a lot of time when customers are waiting for answers and possibly deciding to order from elsewhere. In theory, the CTP models should never become infeasible, because whenever someone initiates a transaction, the model starts with a feasible solution. However, numerical problems related to variable fixing caused difficulties. Once a CTP model accepts an order line item, that line item should remain ordered at the same level until someone reinitializes the model or deletes the order. To guarantee that an order line item does not go short as new line items are added to the solution, the application fixes the variables for unfulfilled demand for committed order line items. It thus creates the potential for fixed basic variables, which may in turn cause problems with numerical infeasibilities. The solution to these problems is twofold: First, we round the unfulfilled demand up to the next biggest integer (or next whole package). Second, instead of fixing the variable to this integer value, we allow it to take on values in a small range around the desired integer value, say plus or minus 0.001. We needed this strategy because increasing the right-hand-side tolerance parameter within the LP solver did not eliminate the numerical infeasibilities caused by fixing basic variables.

We also established procedures for quick recovery and procedures to save information that would help us to diagnose and fix problems. We developed control models for each plant to allow us to monitor and interact with the CTP, scheduling, and ATP models. The CTP and ATP models run as background processes, so the control models allow us to retrieve information interactively. The control models also provide functionality to run the refresh process interactively and to monitor the progress of the refresh process whether initiated from the control models or as part of the standard process.

Basis Management
Occasionally, the solver could not use the advanced basis we were providing from the previous solution, and solution time would be unacceptably long. For each query, we can add variables and constraints, so the basis saved from the previous solution could have fewer elements than the number of rows in the new problem. We greatly enhanced the model’s efficiency by automatically adding any new rows to the advanced basis (and solving with dual simplex). The resulting starting basis is not necessarily feasible, but it provides a better starting point for the solver than the short basis originally provided.

The Available-to-Promise Models
At 15-minute intervals, the ATP models extract the starting inventory, committed orders, and production schedule from their CTP counterparts for each plant and process, creating a snapshot of the current environment. Holding the entire production schedule fixed, a separate LP model calculates the unsold production from each shift by maximizing the quantities of fresh and frozen products not needed to satisfy existing orders, posts this information to the AS/400 database, and updates screens on the CSRs’ desktop.
Because demand for some products is greater than for others and the plants must process each entire animal during a single shift, projected unsold inventory may build up. The sales team needs real-time visibility of this inventory to proactively sell it. While we could have extracted this information from the CTP models, doing so to provide periodic updates would have meant diverting the models from their primary function of solving CTP queries. Consequently, we developed a family of satellite models with an identical time-period structure to provide the information. We run these ATP models on dedicated servers to avoid affecting the response times of the CTP models.

We disable the ATP models when we take the CTP models off line during the twice-a-day refresh process. After we bring the CTP models back on line, we immediately restart the ATP models.

**Implementation Issues and Experiences**

The greatest risks in project implementation typically lie in ensuring the availability or integrity of data and the client organization’s ability to adapt to changes. With a proven technology and project methodology, the technical aspects of a project seldom entail risk. However, in Project Phoenix, the technical challenges were formidable. While creating the components of the application was challenging, getting the application’s many moving parts to work together reliably and efficiently was very difficult.

Six software components had to communicate quickly and reliably. Because Swift uses the application in real time to run its daily sales and production operations, downtime means lost sales. Unfortunately, the complexity of the hardware and software made it impractical to create a test environment that would replicate production stresses, particularly peak volumes of 300 transactions per hour. Our first attempts to go live revealed weaknesses that caused system failures. To mitigate this problem, we recorded transactions from the old system for entire days and wrote scripts to simulate the loads on the system. We installed sniffer programs to track overhead, queuing time, and processing time as queries flowed through the system.

After monitoring the queue length at the CTP engine, we developed a configurable Java-based queue manager. The CTP application consists of a multithreaded server that receives order requests, sends them to the appropriate LP models, receives the answers, and returns them to the requesters. The CTP server performs various numeric and data-integrity calculations and manages the queues for the LP models. The models must process transactions sequentially because each transaction requires re-solving the model unless it contains only line-item deletions. If requests for a particular model arrive faster than it can answer them, they queue up in the CTP server awaiting their turns.

Swift sometimes had queue lengths of six for a single LP model. With solution times of about 20 seconds, the last request in the queue will be in the system for almost two minutes.

The LP models used for CTP have nonlinear solve times with respect to the number of items in the request. It may take 15 seconds to answer a one-line-item query, but only 20 seconds to answer a 40-line-item query. In the worst case, five one-line-items arriving in the queue within 15 seconds would take 75 seconds. If they could be processed together as a batch, it might take only 20 seconds to answer all five requests.

We modified the dispatching process in the CTP server that scans the input queue for requests to examine all requests in the queue for each model and combine them into a single request. After the model processes the request, the application breaks the batch up into its component parts and puts them into the output queue as separate replies. Because not all transaction types can be solved together, we developed a rule-based algorithm to allow the system to identify transactions that can be combined.

The modified queue manager reduced the average number of waiting queries (Figures 4 and 5). For example, of one day’s 290 transactions, 81 were queued without the queue manager in place. The queue manager combined six of those transactions, reducing the number of queued transactions to 75, cutting the total time they spent in queue from 1,624 seconds to 1,115 seconds (509 seconds saved) and reducing the average time spent in the queue from 20.1 seconds to 15.7 seconds and the maximum time from 59 seconds to 40 seconds.
Figure 4: During each day, heavy loads are placed on the system during predictable periods, typically mid to late morning and right after lunch. The high rate of transaction arrivals to the CTP system during these periods increased the average lengths of queues for individual models, increasing system response times before we implemented the queue manager.

Figure 5: After we implemented an advanced queue manager capable of combining similar queries in a single bundled transaction, the average queue length during peak periods dropped. The response time is more sensitive to the number of transactions than it is to the number of line items within each transaction.
Model validation and data gathering and cleansing were daily, iterative processes. Starting with the scheduling model for the Greeley, Colorado plant, a subject matter expert who was intimately familiar with Swift’s beef operations checked the production schedule each day. She identified missing or incorrect data elements, production constraints, and business rules and referred the problems to responsible team members for remediation. The dedication and commitment of an expert with the knowledge of the business needed to guide the data and model validation were essential to the project’s success. By using the scheduling models to validate the model data, we mitigated a major risk before proceeding to the application’s CTP and inventory-availability components.

We spent 16 months and over 25,000 man-hours on Project Phoenix. We put the scheduling, CTP, and ATP models into production for all plants on June 17, 2002. While we were constructing the models, we met with user groups to outline new business processes and to explain methods for defining, using, and interpreting data, for configuring the user interface, and for defining user processes. The new system required users to change the methods they used to do their jobs. Users in accounting, cattle procurement, customer service, industrial engineering, information technology, logistics, plant operations, pricing, production scheduling, sales, and shipping had to make moderate to enormous changes.

**Benefits and Results**

Aspen Technology and Swift & Company jointly conducted a comprehensive postimplementation audit of benefits about 20 months after putting the system into operation. We identified key metrics for business operations and for financial results and analyzed three years of historical data. When data were unavailable, we gathered anecdotal evidence by interviewing stakeholders affected by the system, including employees in operations and scheduling, sales, pricing, warehousing, logistics, and information technology.

We analyzed shipment history from June 2001 through August 2003 by comparing 115,000 records from before Project Phoenix went live with 118,000 records from after its implementation. We found that on-time performance improved. Swift & Company’s business is seasonal, and the improvement is most apparent during the peak demand period that occurs during the summer (largely because of more outdoor grilling), when the percentage of orders shipped on time plus or minus one day increased from 65 to 87 percent. The number of deliveries more than one day early dropped from 11 to seven percent, while those more than one day late dropped from 24 to six percent. We observed improvements in other seasons, although they were less dramatic, suggesting that the system is most helpful when the business complexity and demand increase.

Swift & Company uses the weekly percent-sold position, the percentage of a week’s scheduled production for which it has existing orders, as a key performance metric with which to monitor its business. In the eight weeks following implementation of the Phoenix system, the weekly percent-sold position jumped on average by 22 percent (an additional 100,000+ boxes of beef per week) (Figures 7 and 8). Because the models dynamically adjusted the production schedules to produce products that customers requested, the company allocated more of the available raw materials and capacity to committed orders prior to fabrication. Consequently, the sales team had fewer inventories to push into the marketplace.

Before the introduction of the system, salespeople focused on selling the current week’s production to customers because so much of it was not what the customers really wanted. Many customers waited to place their orders towards the end of the week to take advantage of the discounts offered as incentives to move the finished-goods inventory. These discounts hurt Swift & Company’s profits, and holding excess inventories meant paying for renting no-name trailers to refrigerate and hold the end products.

The application allows Swift & Company to schedule and produce the products its customers actually want. With the system in place, salespeople focus on selling future weeks’ production, not the current week’s. Customers, aware that Swift & Company may actually sell out its production, now place their orders early to insure that they get the products they want. Swift offers discounts less often, improving its margin performance. It produces fewer products for which
there is no demand, decreasing its use of trailers for temporary storage, and their costs, by 90 percent.

Before we implemented the new system, Swift & Company’s unreliable execution caused some customers to stop ordering from it periodically. This tendency of customers to take a break from dealing with Swift has disappeared. The new system is more reliable than the previous system and the company seldom loses orders because the system is unavailable.

The postimplementation audit also showed a reduction in the time CSRs spent on non-value-added activities, such as expediting orders and managing late deliveries. We estimated that 7,280 hours per year were freed for revenue-generating activities, such as pursuing sales goals and expanding the scope of profitable premium cattle programs. Swift & Company has improved its overall competitive position in the industry and is now taking advantage of the increasing business complexity in the marketplace. It is using its new operational and analytical capabilities to take advantage of revenue-generating opportunities with key customers.

The total audited benefits Swift realized in the first year after we delivered the scheduling and CTP application were $12.74 million, broken down as follows:

- Optimized product mix: $12,000,000
- Reduction in orders lost because of system problems: $20,000
- Reduction in price discounting: $560,000
- Reduction in temporarily lost customers: $160,000
The largest component, due to optimized product mix, can be attributed to the system’s dynamically altering the schedule to fill orders as they are entered, rejecting orders that cannot be filled, and producing fewer products for inventory. Further, the system juggles product pieces of varying ages so salespeople can fill orders for immediate shipment. Given the project cost of $6.4 million, this represents a 200 percent return on investment in the first year. Further, the improved sales and operational performance will increase the profitability of Swift & Company’s beef operations for years. Margins in the beef industry typically range from 0.5 to 1.0 percent. The return on this project in the first year represents an increase in earnings before taxes of 13 percent.

Salespersons have accepted new practices. Because CSRs can now see all of the products available, they are no longer allowed to override the system and enter orders that the application determines cannot be filled. Further, they cannot use block orders (orders intended to capture large blocks of future capacity and raw material, which they changed or cancelled as the scheduled production dates approached, disrupting the production schedule and creating unwanted inventory). They must now work to control the percent sold by primal to promote a uniform turnover of inventory.

Managers used to base decisions on anecdotal knowledge, estimates, and guesswork because they had no reliable or complete data. The scheduling and CTP production schedules now provide a solid foundation for analytical decision making. The clear view of sales and operations performance the new system provides has enabled the company to adopt
a new business model (Figure 9), to develop a forecasting and demand-management application, and to focus on procuring cattle for its high-margin premium brands.

Its improvement in forecasting, cattle procurement, and manufacturing have improved Swift & Company’s reputation in the marketplace. Although they increased the complexity of its operations, the company has been able to pursue new opportunities with customers interested in developing long-term relationships. The number of cattle programs has tripled since the implementation of Project Phoenix. Customers are demanding products made to their specifications. The application’s optimization capability is an effective tool for coping with this brand and product proliferation.

In recent years, the beef industry has been affected by a number of external events. The Atkins Diet fad allowed greater control over product pricing, providing better margins. This advantage lasted only until the mad-cow-disease scare in 2003 caused the United States to ban imports of live cattle from Canada, severely reducing the cattle supply. Japan had been a primary export market for Swift & Company, but a case of mad cow disease in the United States provoked a Japanese import ban on American beef products. This ban resulted in Swift having more beef to sell domestically, including some products (for example, beef tongues) that do not have much of a domestic market. This volatility affected customer service and profit margins. The system has provided accurate data on product availability that has been essential in managing prices and margins in this dynamic market while optimally allocating now scarce cattle to customer orders.

Our application of LP technology to solve such large and complex problems in real time would not have been possible five years earlier. Our application
is at the leading edge of what can be accomplished with current computer and solver technology. Aspen Technology and Swift & Company took risks to create it, which paid off in a scheduling and CTP application that has transformed Swift & Company’s beef operations.

Appendix

Model Formulations

Boxed-Beef-Scheduling Models

Sets

- \( O \) cutting operations
- \( P \) parent pieces
- \( P' \) primal pieces \((P' \subset P)\)
- \( C \) child pieces
- \( H \) carcass types
- \( W \) coproducts \((W \subset C)\)
- \( F \) finished products
- \( G \) packaging operations
- \( I_R \) fresh-inventory locations
- \( I_Z \) frozen-inventory locations
- \( I \) inventory locations \((I = I_R + I_Z)\)
- \( T \) time periods
- \( L \) order line numbers
- \( A \) product ages
- \( M \) sets of limited child pieces (by number of pieces per hour)

\( N \) sets of limited finished products (by number of boxes per hour)

Notation

- \( O(i) \) cutting operations that can be performed on \( i \in P \)
- \( P(k) \) parents of \( k \in C \)
- \( h(i) \) carcass yielding \( i \in P' \)
- \( L(i, k, f, t) \) order line numbers in \( t \in T \) for \( f \in F \) from \( i \in I \) accepting \( k \in A \)
- \( A(l) \) acceptable ages for \( l \in L \)
- \( \alpha(t) \) time period in which product must be frozen to be available, \( t \in T \)
- \( \beta(t) \) time period following, \( t \in T \)
- \( \kappa(k, t, u) \) age in period \( u \in T \), where age is \( k \in A \) in period \( t \in T \)

Parameters

- \( \phi_j \) cutting-operation cost for \( j \in O \)
- \( \sigma_g \) packaging-operation cost for \( g \in G \)
- \( \rho_l \) sales price for \( l \in L \)
- \( \tau_l \) unsatisfied-demand penalty for \( l \in L \)
- \( \delta \) capacity-violation penalty
- \( c_{h, t} \) carcass availability for \( h \in H, t \in T \)
- \( a_{j, k} \) yield of child, \( k \in C \), from cutting operation, \( j \in O \)
- \( b_{g, k} \) consumption of child, \( k \in C \), by packaging operation, \( g \in G \)
- \( d_l \) box demand for \( l \in L \)
- \( \theta_i \) demand filled from inventory for \( l \in L \)
- \( \phi_{m, l} \) piece production limit on \( m \in M, t \in T \)
- \( \gamma_{n, t} \) box production limit on \( n \in N, t \in T \)
- \( \lambda_l \) inventory limit at \( i \in I \)
- \( \tau_k \) age indicator \( k \in A, 1 \) for age 0, 0 else
- \( \xi_k \) age indicator \( k \in A, 1 \) for age = number of days needed to freeze, 0 else

Variables

- \( x_{i, j, t} \geq 0 \ \forall \ i \in P, \ j \in O(i), \ t \in T \)
- \( y_{g, t} \geq 0 \ \forall \ g \in G, \ t \in T \)
- \( s_l \geq 0 \ \forall \ l \in L \)
- \( z_l \geq 0 \ \forall \ l \in L \)
- \( \mu_{m, t} \geq 0 \ \forall \ m \in M, \ t \in T \)
- \( \pi_{n, t} \geq 0 \ \forall \ n \in N, \ t \in T \)
- \( \varphi_{i, l} \geq 0 \ \forall \ i \in I, \ t \in T \)
- \( u_{f, i, t} \geq 0 \ \forall \ f \in F, \ i \in I_R, \ t \in T \)
\[
\begin{align*}
\forall f \in F, \quad v_{f,i,t} &\geq 0 \quad \text{frozen box production} \\
i \in I_L, \quad t \in T
\end{align*}
\]

\[
\begin{align*}
\forall f \in F, \quad r_{f,i,k,t} &\geq 0 \quad \text{ending inventory} \\
i \in I, \quad k \in A, \quad t \in T
\end{align*}
\]

\[
\begin{align*}
q_{k,l} &\geq 0 \quad \forall k \in A, \quad l \in L \\
sales from production \\
w_{k,t} &\geq 0 \quad \forall k \in W, \quad t \in T \\
coproduct production
\end{align*}
\]

Minimize
\[
\begin{align*}
&\sum_{i \in P, j \in O(i), t \in T} \psi_{i,j,t} + \sum_{g \in G, t \in T} \sigma_{g,t} - \sum_{l \in L} \rho_l s_l \\
&+ \sum_{l \in L} \omega_l z_l + \sum_{m \in M, t \in T} \delta_{m,t} + \sum_{m \in N, t \in T} \delta_{\bar{\pi},t} + \sum_{i \in I, t \in T} \delta_{\varphi,i,t}
\end{align*}
\]
subject to

carcass availability (ensures all available carcasses are processed):
\[
\sum_{j \in O(i)} x_{i,j,t} = 2c_{ch(t)}, \quad \forall i \in P', \quad t \in T,
\]

child production balance (ensures all produced non-coproduct pieces are either further processed or packaged):
\[
\sum_{i \in P(k), j \in O(k)} a_{j,k,x_{i,j,t}} - \sum_{i \in O(k)} x_{i,k,t} - \sum_{g \in G} b_{g,k} y_{g,t} = 0 \\
\forall k \in C - W, \quad t \in T,
\]

coproduct production (calculates total production of each coproduct):
\[
\sum_{i \in P(k), j \in O(k)} a_{j,k,x_{i,j,t}} - w_{k,t} = 0 \\
\forall k \in W, \quad t \in T,
\]
fresh and frozen box production (forces all produced boxes to go into fresh or frozen inventory):
\[
\sum_{g \in G(f)} y_{g,t} - \sum_{i \in I_L} u_{f,i,t} - \sum_{i \in I_L} v_{f,i,t} = 0 \\
\forall f \in F, \quad t \in T,
\]
fresh-inventory balance (starting fresh inventory + new fresh production – fresh sales = ending fresh inventory):
\[
\forall f \in F, \quad i \in I_L, \quad k \in A, \quad t \in T,
\]
frozen-inventory balance (starting frozen inventory + new frozen production – frozen sales = ending frozen inventory):
\[
\forall f \in F, \quad i \in I_L, \quad k \in A, \quad t \in T,
\]

Demand balance (ensures all demand not fulfilled from inventory is fulfilled from production or shorted):
\[
\sum_{k \in A(f)} q_{k,l} + z_l = d_l - \theta_I \quad \forall l \in L,
\]
total sales (calculates total sales from inventory and production):
\[
s_l + z_l = d_l \quad \forall l \in L,
\]

Piece production limits (enforces any limits on child production):
\[
\sum_{k \in m, i \in P(k), j \in O(i)} a_{j,k,x_{i,j,t}} - \mu_{m,t} \leq \phi_{m,t} \\
\forall m \in M, \quad t \in T,
\]

Boxed-Beef Capable-to-Promise Models

We formulated boxed-beef CTP models very similarly to the boxed-beef scheduling models. The major formulation differences are as follows:

1. Coproduct production is not tracked in CTP models.
2. All sales are from production.
3. There are no inventory locations or associated inventory capacities.

Boxed-Beef Available-to-Promise Models

Sets

\[
\begin{align*}
F &\quad \text{finished products} \\
T &\quad \text{time periods} \\
L &\quad \text{nonforecast order line numbers} \\
A &\quad \text{product ages} \\
A' &\quad \text{product ages greater than 0}
\end{align*}
\]

Notation

\[
\begin{align*}
\alpha(t) &\quad \text{time period in which product must be frozen to be available, } t \in T \\
\beta(t) &\quad \text{time period following, } t \in T \\
\kappa(k, t, u) &\quad \text{age in period } u \in T, \text{ where age is } k \in A \text{ in period } t \in T
\end{align*}
\]
Parameters

- \( u_{f,t} \): box production of \( f \in F, t \in T \)
- \( v_{f,t} \): freezing operations of \( f \in F, t \in T \)
- \( c_{f,k,t} \): fresh order sales from production of \( f \in F, k \in A, t \in T \)
- \( d_{f,k,t} \): frozen order sales from production of \( f \in F, k \in A, t \in T \)
- \( \mu \): number of days needed to freeze
- \( \xi_k \): age indicator \( k \in A, 1 \) for age = \( \mu \), 0 else

Variables

- \( x_{f,t} \geq 0 \ \forall f \in F, t \in T \): available fresh production
- \( y_{f,t} \geq 0 \ \forall f \in F, t \in T \): available frozen production
- \( r_{f,k,t} \geq 0 \ \forall f \in F, k \in A, t \in T \): ending fresh inventory
- \( s_{f,k,t} \geq 0 \ \forall f \in F, k \in A, t \in T \): ending frozen inventory

Maximize \( \sum_{f \in F, t \in T} x_{f,t} + \sum_{f \in F, t \in T} y_{f,t} \)

subject to

- fresh-production balance (calculates available fresh production in each time period as fresh production in that period – freezing operations – fresh sales – ending fresh inventory): \( x_{f,t} + r_{f,0,t} = u_{f,t} - v_{f,t} - c_{f,0,t} \ \forall f \in F, t \in T \)
- fresh-inventory balance (starting fresh inventory – sales = ending fresh inventory): \( r_{f,k,t} - r_{f,k(\beta(t),1)} = c_{f,k,t} \ \forall f \in F, k \in A, t \in T \)
- frozen-inventory balance (calculates available frozen production in each time period as newly frozen product in that time period + starting frozen inventory – frozen sales – ending frozen inventory): \( s_{f,k,t} - s_{f,k(\beta(t),1)} - \xi_k y_{f,k,t} = d_{f,k,t} - \xi_k v_{f,0(t)} \ \forall f \in F, k \in K, t \in T \)