

# Intelligent Retail Logistics Scheduling

*John Rowe, Keith Jewers, Joe Sivayogan,  
Andrew Codd, and Andrew Alcock*

■ The supply-chain integrated ordering network (SCION) depot-bookings system automates the planning and scheduling of perishable and nonperishable commodities and the vehicles that carry them into J. Sainsbury depots. This initiative is strategic, enabling the business to make the key move from weekly to daily ordering. The system is mission critical, managing the inward flow of commodities from suppliers into J. Sainsbury's depots. The system leverages AI techniques to provide a business solution that meets challenging functional and performance needs. The SCION depot-bookings system is operational, providing schedules for 22 depots across the United Kingdom.

**J.** Sainsbury is the United Kingdom's most well-established retailer, with a market share of 11.7 percent of the United Kingdom's food retail market and group annual sales of £12 billion (US\$19,776,000,000) (financial year 1995). J. Sainsbury has extensive assets, with subsidiaries such as Shaws in the United States and the Savacentre and Homebase chains in the United Kingdom.

Given J. Sainsbury's position in the retail market, the efficient and effective running of the supply chain for J. Sainsbury is critical to the mission of the organization. The J. Sainsbury logistics purpose statement is

to manage the flow of goods from supplier to shelf, ensuring that the customer has the right product in the right place at the right time.

To these ends, J. Sainsbury's Logistics Group is committed to being world class. The group's direction principle is to be seen as the world's

best logistics team. In line with the logistics mission, a strong focus has been on developing a supply chain that leads the field in terms of providing highest-quality service to the customer while reducing operating costs.

The Supply-Chain Integrated Ordering Network (SCION) Project is an element in the reengineering of the J. Sainsbury supply chain. SCION reengineers the ordering and booking processes of the depot-replenishment links of the supply chain for perishable and nonperishable commodities. This move is from a vertical to a horizontal supply chain. The SCION depot-bookings system is a critical link in this chain because it is positioned with the forecasting and ordering links to its left and the distribution, warehousing, and supplier links to its right in the supply chain.

The SCION depot-bookings system is categorized as a strategic enabler to allow the business to move from weekly to daily ordering. The business advantages of this move are reduced stock levels and greater flexibility in the placement of orders. Daily ordering enables the business to run with less stock in the supply chain yet provides higher levels of customer service. Daily ordering is the adoption of the idea from the manufacturing industry of *just-in-time processing*, that is, making material available for a value-adding activity in a process at the point in time it is required—not before or after.

The SCION depot-bookings system is a business-process automation system. The system automates the vehicle-planning and vehicle-scheduling processes.

The SCION depot-bookings system is required to run for 22 warehouses and process between 100,000 and 200,000 pallets of nonperishable commodities to be placed on 5,000

to 10,000 vehicles an order cycle. The system is required to run in an operational window of two hours.

## Application Description

In this section, we discuss the purpose of the system; the determine-vehicle-contents, the determine-day-of-delivery, and determine-booking-time processes; and performance requirements.

### Purpose

Under the daily ordering regime, an *order to a supplier* is defined as a vehicle with contents for a specific delivery time to a receiving slot on a shift of a depot. An *order* is simply a purchase order for a quantity of a commodity. The depot-bookings system produces a schedule of orders for each of the 22 warehouses. The orders are sent by electronic data interface (EDI) protocols to J. Sainsbury's several thousand suppliers on the day of the bookings run.

The time scales under daily ordering are too restrictive for the business to have a global view of the schedule because the time window for viewing the schedule is three hours. The volume of vehicles and their commodities is high—as many as 10,000 vehicles and 200,000 pallets of commodities. These volumes have led the business down the process-automation route to produce the depot-bookings system. This gives the business control over the vehicles and contents by placing orders daily. Daily ordering gives the benefits of a more responsive supply chain and reduced manual intervention. It also enables stock cover to reds. The schedule has a different view depending on where in the business it is examined. The schedule can be conceived in business terms as the composite of all the orders for a depot sent to suppliers as the result of a daily depot-bookings run. Alternatively, the schedule can be viewed as all the vehicles going into a depot on any given day in response to the vehicle, its contents, and the delivery time generated by the depot-bookings system. From a logistics controller's perspective, the schedule is those vehicles and commodities that belong to the suppliers that they manage.

### Determine Vehicle Contents

The *determine-vehicle-contents process* takes orders for commodities in the form of delivery units. A *delivery unit* is a pallet or a part pallet of some commodity. The requirements of this process are as follows: (1) all delivery units on

a vehicle belong to the same supplier or are transported by the same hauler; (2) the possible delivery days of each delivery unit assigned to the vehicle have some overlap; (3) there is a good mix of products on each vehicle; (4) the volume and the weight of the delivery units assigned to a vehicle do not exceed the vehicle capacities; (5) part-pallet delivery units are aggregated into full pallets, provided they are from the same supplier; (6) vehicle fills are balanced while the number of pallets delivered before their ideal delivery date is minimized; (7) existing vehicles are topped up before new vehicles are created; and (8) the minimum number of vehicles is used.

### Determine Day of Delivery

Once the contents of a vehicle have been determined, a day of delivery is assigned to each vehicle. This process takes into account (1) the depot capacities for the week in terms of pallets and vehicles, (2) the possible delivery days of each supplier and hauler, (3) the possible delivery days of each vehicle, (4) the spread of vehicle load sizes across the week, and (5) the spread of suppliers' and haulers' deliveries across the week.

### Determine Booking Time

This process assigns a booking time to as many vehicles as capacity allows using supplier-hauler delivery-time preferences, the supplier-hauler's imperative to the business, the spread of vehicle load sizes across the day, and depot shift and receiving-slot capacities in vehicles and pallets.

### Performance Requirement

Because of the strategic goal of the system to enable daily ordering, the SCION depot-bookings system has to run in a restrictive time window of two hours. This time window is determined by the business's operational timetable and, as such, is a hard requirement. The system is required to process the order of 100,000 delivery units, which means building and scheduling about 7,000 vehicles while observing the functional requirements stated previously.

## The Software-Solution Architecture

The system has a three-layer architecture, shown in figure 1. The top layer of the system manages program flow. The middle layer, the *solution level*, consists of designed subprocesses that perform metalevel processing over the

*The approach taken was to model the business domain as classes of objects with relationships between classes.*

model of the domain. The third layer is a model of the business domain.

The top level of the system consists of forward-chaining rules that govern program flow. Pattern matching is used to determine subprocess end points. When a given subprocess has completed, the top level of the system fires a rule that sends a message, causing the next subprocess to be performed.

The middle layer consists of solution service providers and subprocess objects. Solution service providers are subsystems or stand-alone classes that perform a well-defined role in the generation of the solution. For example, the *best-of-type class* is an abstract superclass that has, as its role, the determination of the best slave object in a master-slave object pattern.

An example of a master-slave(s) relationship in the domain is the multiple-cardinality relationship between suppliers and their vehicles. A supplier will have many vehicles. The best-of-type class has knowledge of the interfaces of the master class and the slave class. The best of type, that is, the best slave in the master-slave relationship, is determined by a method for the specific kind of best-of-type class. In the supplier-vehicle instance, we might be interested in the biggest vehicle in terms of weight or volume, or we might be interested in the best of type in terms of the attachment of priority that the business places on the contents. Best of type is particularly useful when considering compound properties of the slave class with multiple-slave instances.

*Subprocess objects* are typically specialized instances of a process-manager abstract superclass. Specialization consists of the knowledge of the representation of the problem domain and any methods required to provide the respective subprocess's services.

The approach taken was to model the business domain as classes of objects with relationships between classes. For example, a depot in the business is modeled as the depot class, a depot's work shift is modeled as a shift class, and a depot receiving slot has a depot shift class. Part of an object model is shown in figure 2, and a class hierarchy is shown in figure 3. The classes in the business-object model provide services modeled on the kind of information that is available about the real-world equivalents of the objects. For example, a receiving-slot object could be sent a message asking what its pallet capacity is; the receiving slot would return the capacity.

Wherever possible and where appropriate, the internal consistency of an object that is

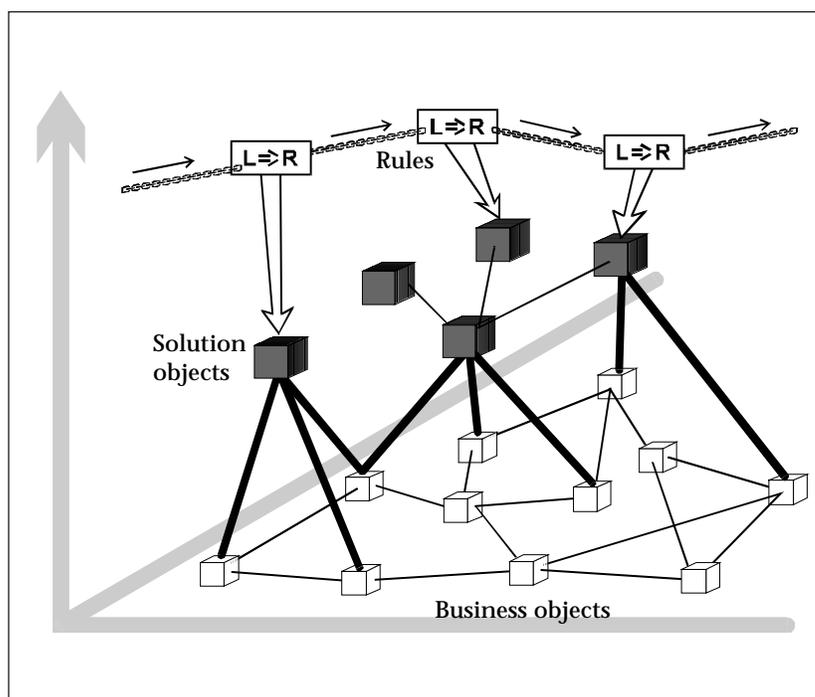


Figure 1. Three-Layer Architecture.

dependent on related properties of the object is maintained in a lazy way. The use of objects to model the business domain gave us a representation that allowed us to build a referentially transparent model of the objects in the domain in terms of the objects' interfaces or services. Similarly, the use of objects enabled us to implement strategies such as lazy evaluation for changes in dependent properties of an object.

## AI Techniques Used

AI techniques are used throughout the system. They are leveraged most heavily in the determine-booking-time subsystem. In the determine-vehicle-contents and determine-day-of-delivery subsystems, the principal techniques used have been the representation and integration of rules and objects. Rules are used to manage the flow from subprocess to subprocess, and object orientation is used to implement the subprocesses and model the business enterprise.

The forward chaining of the production rules manages the process flow of the determine-vehicle-contents subsystem. When the state of the business-object model indicates that there is no process currently operating, a production rule fires and initiates the next

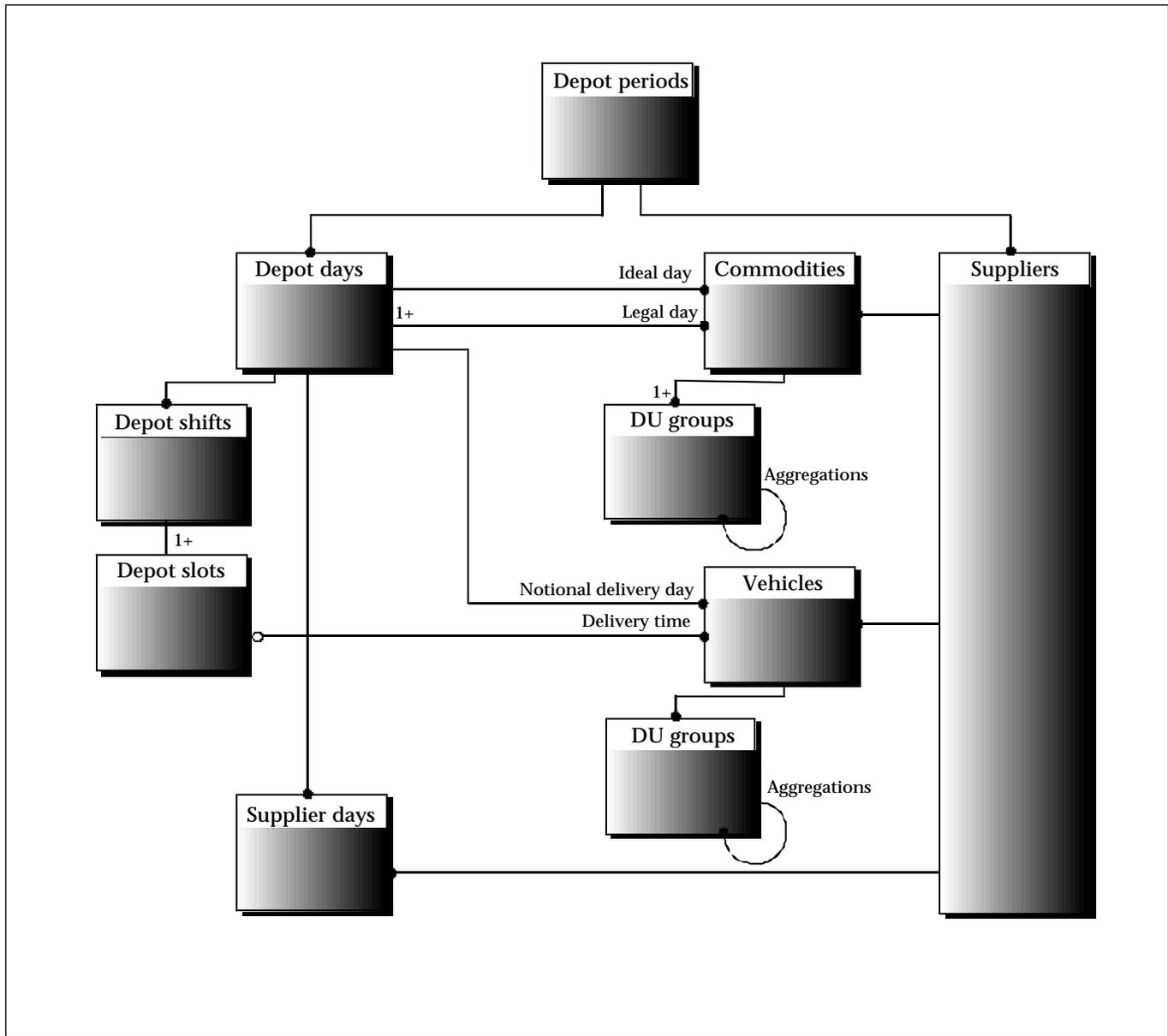


Figure 2. Object Model.

process in the bookings run by sending a message to the object responsible for the process. Over the business-object model, four key processes drive the generation of booked vehicles for a day: (1) targets, (2) best of type, (3) assignment, and (4) constraint propagation.

### Targets

The *target process* manages all permissible supplier deliveries to all receiving slots on all shifts on a day. The principle of least over-

commitment is a scheduling heuristic for putting filler objects into multiple container objects. The principle of least overcommitment for a set of containers and a set of fillers, where each filler can go into some, but not all, of the containers, is (1) calculate, for each container, the number of fillers that can go into it; (2) pick the container with the smallest number of fillers that can go into it, and put a filler into it; and (3) repeat steps 1 and 2 until there is either no space in the

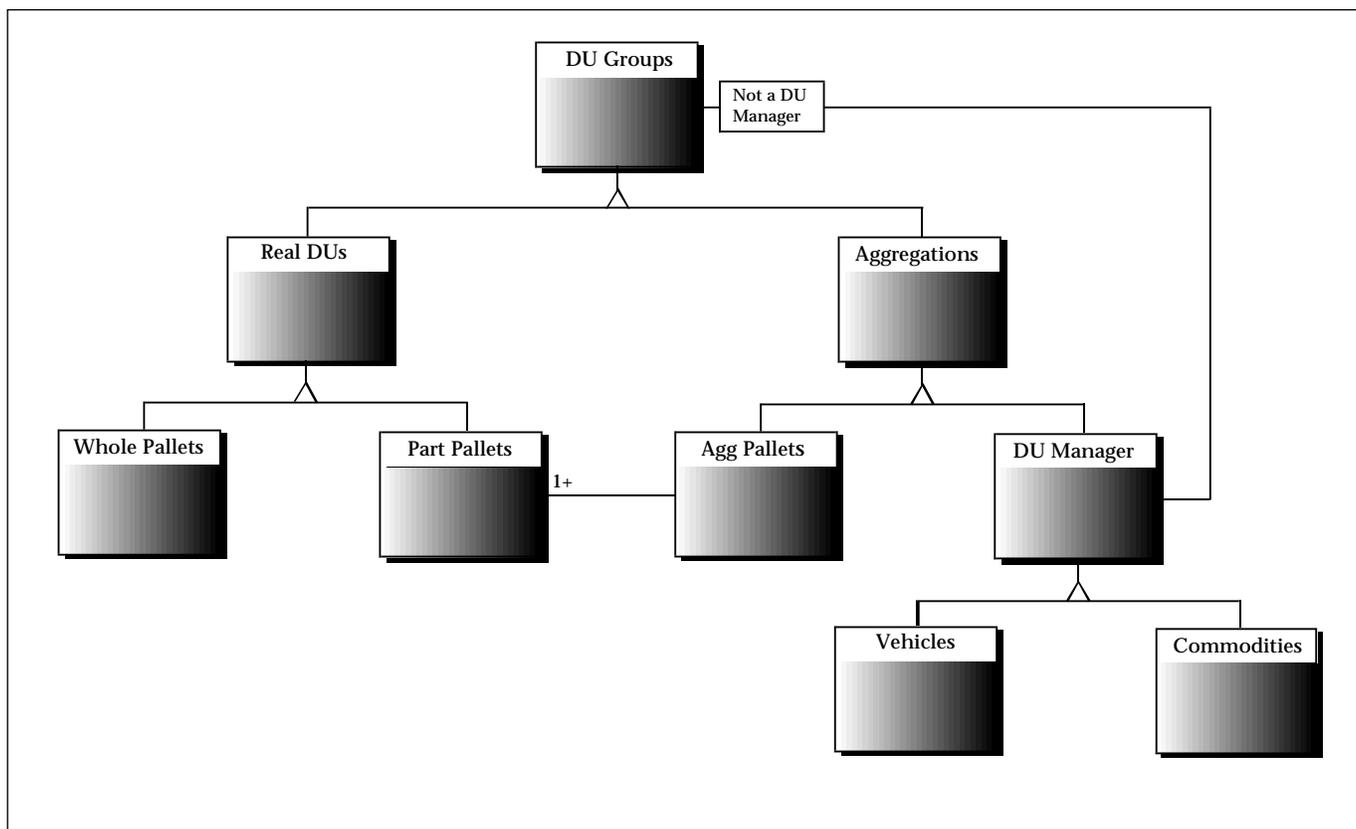


Figure 3. Class Hierarchy.

containers for the fillers or no fillers left.

Contrast this scenario with just naively putting fillers into containers. If we put fillers into containers simply on the basis of where the fillers would ideally like to go, then we run out of capacity for the most overcommitted containers and consume fillers that were permitted to go into less overcommitted containers. This results in the most overcommitted containers being full and the remaining containers unable to be filled because they are not permissible for the fillers remaining. A supplier has a range of preferred delivery times. This range is expressed by relationships from the supplier to the slot. There is a relationship for each preference for a receiving slot. The graph of all relationships to slots expresses all permissible deliveries into the depot on the day. A supplier will have a most preferred delivery time and a least preferred delivery time. A weighting algorithm proportionally distributes the number of pallets on vehicles with respect to suppliers' preference for a delivery time. This information is termed the *commitment value*. The commitment value

is stored for the relationships between the supplier and the receiving slots. The relationships are represented as linking objects. For a given receiving slot, we can access all the permissible linking relationships associated with the slot, which means we can derive a value for the commitment for the receiving slot by aggregating deliveries with the commitment values on them of the links into the receiving slot. The aggregation of commitment values on the linking objects into a slot gives us the commitment value for the receiving slot. The commitment value for a shift is the aggregation of slot commitment values.

Commitment only provides us with information about permissible deliveries to the depot. It does not give us any information about how the depot capacities are configured on the slots and shifts. This information is built in by dividing the commitment for the slot or shift by the capacity of the slot or shift. This value is termed the overcommitment for the slot or shift. The target process sets up the infrastructure so that the least overcommitted shift can be determined.

## Best of Type

Best of type uses a fitness function to determine the best child in a one-to-many parent-child relationship. The children are all the same abstract type. Best of type provides meta-level information about the business-object model. Best of type is used when the receiving slot is the parent, and the preference links are the children; the shift is the parent, and the slots are the children; and the depot day is the parent, and the shifts are the children.

The fitness function for the best-preference link finds the best link according to the following criteria: preference, supplier priority, commitment, and vehicle size. The fitness function for the best slot uses the best of type for the link, overcommitment, and a weighting for assignments to each slot so far. The fitness function for the best shift uses the best slot and overcommitment for the shift. The criteria for the best of type were elicited from the business experts and refined through a prototyping process. Best of type is implemented as an abstract superclass where specific best of types are specializations of the best-of-type class.

## Assignment

Assignment is implemented as the rule *focus decision demand spreading*. The RETE algorithm manages the rule firing. The condition of the rule is the best of type for the relationship between the day and its shifts. The action of the rule is to traverse the business-object model, finding the best shift's best slot; the best slot's best link; and, thus, the supplier. The supplier object provides the service of best vehicle. The *best vehicle* is the biggest vehicle that will fit within the constraints of the slot and shift capacity remaining for the best slot and shift. This implements the packing heuristic of always placing the biggest fillers in a container before the smaller fillers. The best vehicle is assigned to the slot and given the slot's opening time as its time of delivery. The vehicle is written out to a flat file to be updated to the database in a subsequent process.

## Constraint Propagation

The business-object model comprises related subsystems of objects. If a change is made to an object, any other class of object with dependent values must be modified. For example, if a vehicle is assigned to a slot, then the slot's pallet and vehicle capacities are consumed. However, the slot's capacities are dependent on those of the shift, and vice versa. Because the value of the assignment is debited from the slot, we need to keep the object model consistent and debit the vehicle pallets

from the shift's capacity and a vehicle from the shift's vehicle capacity. Similarly, in an assignment, the link from the supplier to the slot will close, entailing the recalculations of the commitment values, overcommitment values, and best of type.

The concept underlying constraint management using process classes is to encapsulate a process in an object. In this context, a process is understood as a sequence of operations toward some specific goal, where the operations are distributed over the business-object model. Complex processes can be built up by a composition of processes. To do this building, a mechanism is required that enables the process to traverse the relationships between the co-dependent objects. These relationships will either be one-to-one relationships or one-to-many relationships. In traversing the business-object model, the process object must be able to access the services of the business object and process the results of the accesses relative to the process's goal. This is implemented by an engine that traverses the object model and a knowledge base of the classes and services that the engine must process. The engine and the template for the knowledge base are services of an abstract-process class. The *abstract-process class* with these services is the base class of any processes that are distributed across the business-object model. The specialized process classes contain knowledge of their environment in terms of the map of classes and operations and an intelligent traversal engine that allows the business-process object to traverse the business-object model while it considers the state of the object model and does not traverse dead paths through the object network. The process objects implement constraint propagation, thus ensuring global consistency across the object model.

## Booking for a Day

The constraint-process objects are triggered by the assignment of a vehicle to a slot. They are sent a message about the nature of the assignment and traverse the object model, ensuring that the receiving slot and shift capacities are modified accordingly, receiving slots and shifts are closed if there is no more capacity available, and the supply group's links are constrained if there are no more vehicles to assign or if there is no capacity on the slots to which they are linked. When the processes have completed their modifications to the business-object model, the targets are recalculated, best of types are reprocessed, and the assignment rule refires. The cycle of con-

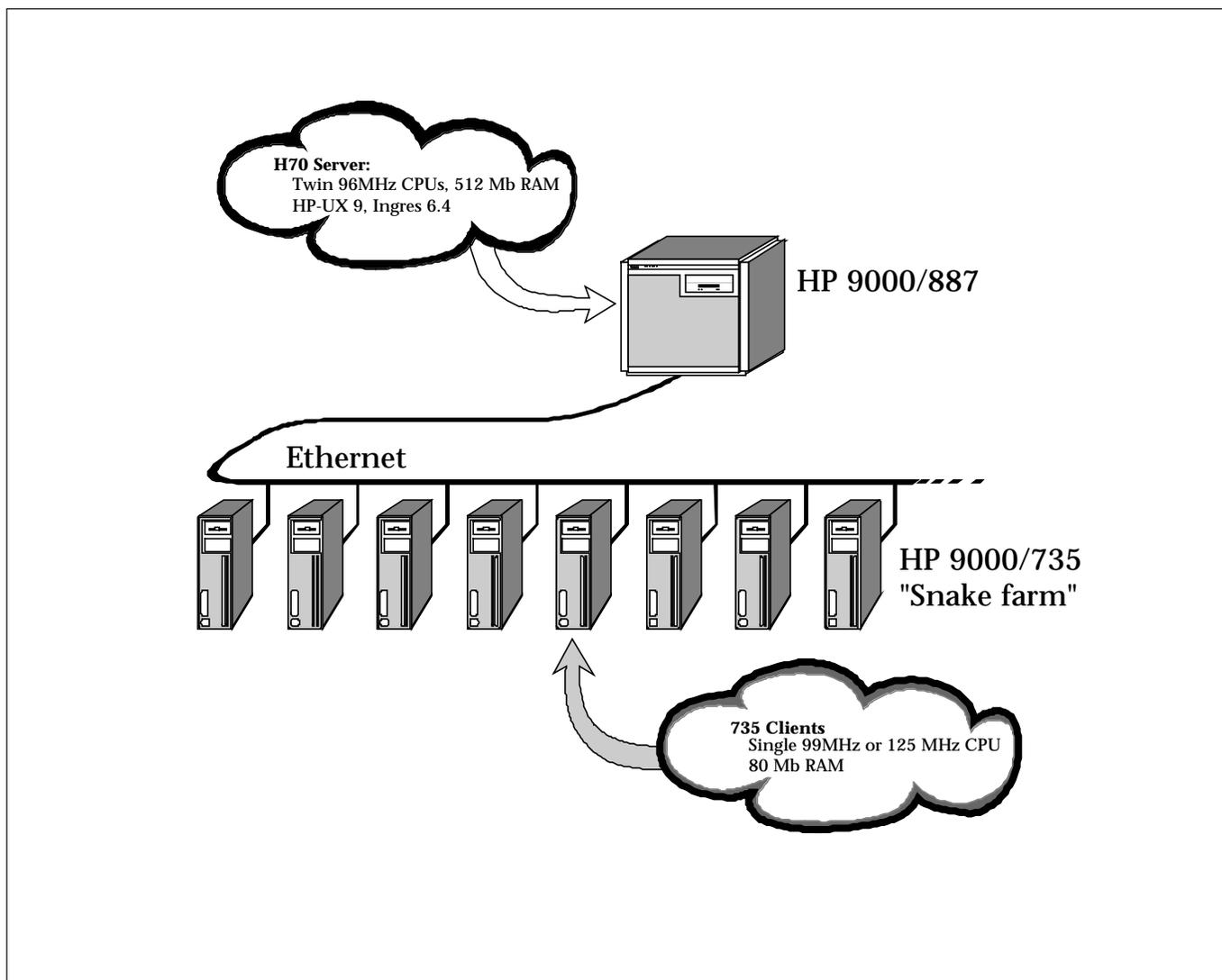


Figure 4. Hardware Configuration.

straint processes, targets, best of type, and assignment continues until no capacity is left on the slots and shifts for permissible assignments, or no vehicles are left to assign. The output of this sequence of processes is a flat file of vehicles with contents and booking times. This file is subsequently loaded into a relational database, and later that day, the orders as vehicles with contents and delivery time are sent by EDI to the suppliers.

### Hardware and Software Environment

SCION depot bookings is written in ART-IM 2.5 R2 on the HP-UX 9 operating system, with an in-

terface to INGRES written in C. SCION runs in a group of HP-UX UNIX machines that make up one H70 server and eight or more HP 9000/735 clients. This configuration is known as a *snake farm* and is shown in figure 4.

The snake farm shares an NFS directory, held on the server, across an Ethernet. The central data repository is an INGRES 6.4 relational database management system held on the server that can be queried by applications running both on the server and also on any client.

The H70 is a twin central processing unit (CPU) mid-range machine with 512 megabytes (MB) of random-access memory (RAM), optimized as a server; the 735s are smaller, single-CPU machines with 112 MB RAM, op-

timized for processor speed. HP's TASK BROKER job-scheduling program is used to distribute jobs efficiently between clients and server and manage the resources of the snake farm.

The SCION depot-bookings operational data are chunked by depot, allowing parallelism across the clients during the two-hour window. TASK BROKER controls the order in which depots are run and best distributes the depots across the snake farm. This allows the SCION depot-bookings scheduling solution to easily be scalable for different data volumes.

## Applications Innovation and Business Significance

The system can claim innovation in the following ways: (1) It uses AI techniques to automate time-constrained business processes. (2) It integrates rule-based, object-oriented, and relational paradigms and leverages AI approaches to provide a business solution. (3) It is an enabler for J. Sainsbury's business strategy. (4) It is mission critical to J. Sainsbury's business. (5) It is a key component in a reengineered business process. (6) The system processes as many as 200,000 delivery units, producing 10,000 vehicles for 22 depots. (7) The system runs in a two-hour time window and resolves a complex task and processes large volumes of data. (8) The system's technical architecture exploits concurrency to perform its function.

## Project History

The life cycle of the project can be divided into three phases: (1) development, (2) continuous improvement, (3) and maintenance.

The development cycle took place between spring 1993 and spring 1994. An evolutionary model was adopted for system development. This process was managed using a time-box approach for each stage, each stage producing a system deliverable. The key system deliverables were (1) conceptual demonstrator, May 1993 to June 1993; (2) prototype, June 1993 to February 1994; (3) production prototype, March 1994 to June 1994; and (4) production system, July 1994 to November 1994.

Each deliverable was seen as a stage in the evolution toward a solution that met all the business requirements. The primary driver for this approach was the management of business risk. In addition, the view of the development team was that all the successful complex systems that they were aware of had been grown over time as opposed to developed using a big-bang approach. At each

stage, J. Sainsbury management had a tangible software deliverable that it could assess and that it could offer feedback to prior to moving forward.

The software development approach was cyclical for each stage. The software development cycles consisted of domain knowledge acquisition, business analysis, solution design, software logical and physical design, incremental build, and expert verification.

The evolutionary aspect of the life cycle was implemented by significant design and code reuse between each stage of the life cycle and the filtering out of approaches and mechanisms that were inefficient or nonrobust or engendered high coupling. Considerable emphasis was placed on creativity in the design periods of a cycle.

The continuous-improvement phase of the development cycle ran between January and November 1995 under a change-management regime. The key driver for change for the system was the radical business-process reengineering that occurred at organizational and process levels at J. Sainsbury. This manifested itself in requirements for enhancements, tuning of the quality of existing functions, and a drive to reduce the system's run time. During this period, the run time was cut in half, from two hours to one hour.

The support phase of the system began in January 1995 and is ongoing for the life of the system. A two-tier model was adopted for the support of the system, consisting of primary support performed by J. Sainsbury and secondary support performed by Inference.

Primary support comes into play if there is a system crash. The input data are manipulated at the database by SQL to remove the errant data that have caused the crash. The system is then restarted and run through to completion. Primary support requires no knowledge of the system's internal design or coding. An error-recovery document contains the necessary knowledge and processes to resolve primary support problems. This document is supported by custom diagnostic tools.

Secondary support comes into play if a system crash cannot be resolved by manipulating the data. This support entails accessing the system at the code level, which is performed by Inference staff with the requisite technical skills set and knowledge of the application's design and coding.

It should be noted that the system is robust. The gearing of the development and implementation approaches has been such to ensure robustness. Evidence of this is that the SCION depot-bookings system dealt with a 50-

percent increase in data volumes over Christmas 1995 and ran within the operational time window. Nevertheless, because of the mission-critical nature of the application, every effort has been made to put in place a practical workable support strategy.

Functional enhancements to the system are made by Inference consultants. The system has been engineered to be extensible. The use of object-oriented approaches supports loose coupling within business-object model and subprocess layers of the architecture. Similarly, the layers themselves are loosely coupled. Additional processes can be inserted into the system flow and subprocess layers of the system by adding rules or creating a specialized process class from the abstract superclass. Because the business-object model represents business reality, the business classes can evolve without jeopardizing the internal structural coherence of the system.

## System Validation

The functional requirements of the application were validated through three distinct processes: (1) testing by the J. Sainsbury business experts within the project team, (2) user-acceptance testing by the business, and (3) receiving feedback following the incremental rollout of the system.

The first process consisted of two months of business-rule verification. During this period, all conceivable operational scenarios were constructed by the business experts against which the system would be validated. The system was tuned where necessary for quality of results. When this process was finished and signed off, the system was handed over to the business for user testing.

User testing started on one depot. The Logistics Group ran the system for one month in parallel with existing procedures, validating the results. Once a level of confidence in the system was gained, the system was incrementally rolled out one depot at a time until confidence was such that large numbers of depot could go live in one hit.

During this period, the business critically evaluated the system's output. In conjunction with operational readiness, the quality of the output enabled the business to judge the speed of system rollout. This approach enabled the business to derive business benefit while it built confidence in the results of the system. The performance requirement of the system was validated by two benchmarking exercises. These exercises consisted of running the system against peak-production data

volumes on a production-configured operating environment.

The robustness requirement of the system was validated by a stress-testing exercise, which consisted of taking production data and randomizing the data variables in their respective valid ranges. This process is ongoing because it periodically yields data conditions that throw the system. These data conditions are trapped and their resolution incorporated into the system.

## Application Deployment and Use

The SCION project was implemented in two phases: (1) the automation of depot bookings under a legacy weekly-ordering system and (2) the migration of the new SCION ordering system that operates on a daily basis.

The goal of the first phase—to move all 22 depots onto the automated depot-bookings process—was achieved by November 1995 (the first depot went live in October 1994). The rollout averaged three new depots moving to the SCION depot-bookings system each calendar month.

The second phase is now complete and represents a radical change in existing operating procedures and processes. Currently, six depots are running under the daily ordering regime.

## Application Payoff

The system is a strategic enabler. As such, the primary benefits of the system are realized across the whole of the supply chain. This occurs with the integration of the other key systems development programs and process reengineering that the J. Sainsbury Logistics Group is engaged in. Nevertheless, it is projected that the SCION project, consisting of SCION ordering and bookings, will produce benefits of more than £10 million (US\$ 16,480,000) in the next 5 years and return on investment in 6 months. This is primarily in the ability to improve the management of stock in the supply chain and improve customer-service levels at the depots. Current stock levels and customer-service levels as a result of SCION over the last year support these projections.

The other key benefits of the system are (1) a reduction in the amount of administration required to manage depot bookings both at the head office and for J. Sainsbury's suppliers; (2) an improvement by the bookings system in the use of depot receiving resources; (3) en-

**... it is projected that the SCION project,  
consisting of SCION ordering and bookings,  
will produce benefits of more than £10 million  
(US \$16,480,000) in the next  
5 years and return on investment  
in 6 months.**

hanced maintenance facilities for managing depot-receiving capacities; (4) support for new concepts critical to the reengineering of the supply chain; and (5) control over the contents of vehicles, hence supporting the management of transport costs.

### Summary

The SCION depot-bookings system automates the planning and scheduling of perishable and nonperishable commodities and the vehicles that carry them into J. Sainsbury depots. This initiative is strategic, enabling the business to move to daily ordering. The system is mission critical, managing the inward flow of commodities from suppliers into J. Sainsbury's depots. The system provides J. Sainsbury with control over the vehicles and goods coming into its depots. The bookings system is written in ART-IM and makes extensive use of AI techniques that are used to provide the business with a solution that meets challenging functional and performance needs. The SCION depot-bookings system is operational, providing schedules for 22 depots across the United Kingdom.

### Bibliography

- Alexander, C. 1979. *The Timeless Way of Building*. Oxford, United Kingdom: Oxford University Press.
- Hammer, M., and Champy, J. 1993. *Reengineering the Corporation: A Manifesto for the Business Revolution*. New York: Harper Collins.
- Hart, A. 1989. *Knowledge Acquisition for Expert Systems*. Reading, Mass.: Addison-Wesley.
- Rumbough, J.; Blaha, M.; Premerlani, W.; et al. (1991). *Object-Oriented Modelling and Design*. Englewood Cliffs, N.J.: Prentice-Hall.
- Winston P. H. 1992. *Artificial Intelligence*. Reading, Mass.: Addison-Wesley.



**John Rowe** has been logistics director at J. Sainsbury plc since October 1994. He is responsible for the strategy, planning, and management of the flow of goods from supplier to shelf, including the development and implementation of computer systems to support the business process. He has held a number of positions during the last 10 years, including head of Group Internal Audit and project manager for a strategic IT system focused on improving supply-chain performance. Prior to joining J. Sainsbury, he studied mathematics at Warwick University and then started his career in accounting with Cooper & Lybrand.

**Keith Jewers** has worked on the development and implementation of a variety of systems within the retail, marketing, trading, and logistics areas of J. Sainsbury, most using technology that was considered leading edge at the time of development. He began in programming and analysis, but his more recent assignments have all been as development manager. He assumed the overall management of the SCION Project in December 1994 when the project was at its height (consisting at the time of a team of 70 people of both IT professionals and secondees from the business). He then took it through the implementation of the bookings and orderings phase (completed in mid-1996). He is currently the development manager of a strategic project that will complement the overall marketing initiative being undertaken by Sainsbury's.

**Andrew Alcock** graduated from Cambridge University in 1991. He worked at the National Grid Company developing client-server applications. He joined Inference in October 1993 and worked on the SCION Project until its completion in June 1996.

**Andrew Codd** graduated from the University of London in 1987. He worked as a knowledge engineer for Fides Software and then for BT, introducing knowledge-based software into its logistics operation. He joined Inference in July 1993 and worked as the Inference project manager for the SCION project until March 1996. He now works as United Kingdom operations manager for Inference International.