Intelligent Retail Logistics Scheduling

John Rowe, Keith Jewers

Office.RoweWJ, Keith.Jewers@js.btx400.co.uk
J. Sainsbury plc,
Stamford House,
Stamford Street,
London, SE1 9LL,
United Kingdom.
+44 (0) 171 921 6000
(Fax) +44 (0) 171 921 6178

Andrew Codd and Andrew Alcock

acodd, aalcock@inference.co.uk
Inference Corporation,
258 Bath Road,
Slough, Berkshire,
SL1 4DX,
United Kingdom.
+44 (0) 1753 771100
(Fax) +44 (0) 1753 771101

Abstract

The Supply Chain Integrated Ordering Network (SCION) Depot Bookings system automates the planning and scheduling of perishable and non-perishable commodities and the vehicles that carry them into J. Sainsbury depots. This is a strategic initiative, enabling the business to make the key move from weekly to daily ordering. The system is mission critical, managing the inwards 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 UK.

Problem Description

Business Context

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

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 organisation. 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, there has been a strong focus on developing a supply chain that leads the field in terms of providing highest quality service to the customer whilst 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 non-perishable commodities. This is a move from a vertical to a horizontal supply chain. The SCION Depot Bookings system is a critical link in this chain as 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.

Objectives of the Application

The SCION Depot Bookings system is categorised as a strategic enabler to enable 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 provide higher levels of customer service. Daily ordering is the adoption of the idea from the manufacturing industry of Just In Time processing, i.e., making material available for a value adding activity in a process at the point in time it is required – not before nor after.

The SCION Depot Bookings system is a Business Process Automation system. The system automates the vehicle planning and scheduling processes.

The SCION Depot Bookings system is required to run for 22 warehouses and process between 100,000 and 200,000 pallets of non-perishable commodities to be placed on 5,000 to 10,000 vehicles per order cycle. The system is required to run in an operational window of two hours.

Application Description

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. This should be contrasted with an order simply being 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 via Electronic Data Interface protocols (E.D.I.) to J. Sainsbury's several thousand suppliers on the day of the bookings run.

The timescales under daily ordering are too restrictive for the business to have a global view of the schedule. This is because the time window for viewing the schedule is three hours. The volume of vehicles and their commodities is very high – up to 10,000 vehicles 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.

The schedule has a different view depending on from where in the business it is viewed. 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 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 part pallet of some commodity. The requirements of this process are:

- All delivery units on a vehicle belong to the same supplier or are transported by the same haulier
- The possible delivery days of each delivery unit assigned to the vehicle have some overlap
- There is a good mix of products on each vehicle
- The volume and weight of the delivery units assigned to a vehicle do not exceed the vehicle capacities
- To aggregate part pallet delivery units into full pallets, provided they are from the same supplier
- To balance filling vehicles with minimising the number of pallets delivered before their ideal delivery date
- Existing vehicles are 'topped up' before new vehicles are created
- 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:

- The depot capacities for the week in terms of pallets and vehicles
- The possible delivery days of each supplier and haulier
- The possible delivery days of each vehicle
- The spread of vehicle load sizes across the week
- The spread of suppliers' and hauliers' deliveries across the week.

Determine Booking Time. This process will assign a booking time to as many vehicles as capacity will allow using:

- Supplier/haulier delivery time preferences
- Supplier/haulier's imperative to the business
- The spread of vehicle load sizes across the day
- Depot shift and receiving slot capacities in vehicles and pallets.

Performance Requirement. Due to the strategic goal of the system to enable daily ordering, the SCION Depot Bookings system has to run in a very restrictive time window of two hours. This time window is determined by the business' operational timetable and as such is a hard requirement. The system is required to process of the order of 100,000 delivery units. This constitutes building and scheduling about 7,000 vehicles whilst observing the functional requirements stated above.

Software Design

The Software Solutions Architecture. The system has a three layered 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 meta-level processing over the model of the domain. The third layer is a model of the business domain.

The top level of 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 super-class 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 slave class. The 'best-of-type', that is the best slave in the master-slaves relationship, is determined by a method for the specific kind of best-of-type class. In the supplier-vehicles instance we may be interested in the biggest vehicle in terms of weight or volume, or we may 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 slaves instances.

Subprocess objects are typically specialised instances of a process manager abstract super-class. Specialisation consists of the knowledge of the representation of the problem domain and any methods required to provide the respective subprocess' services.

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

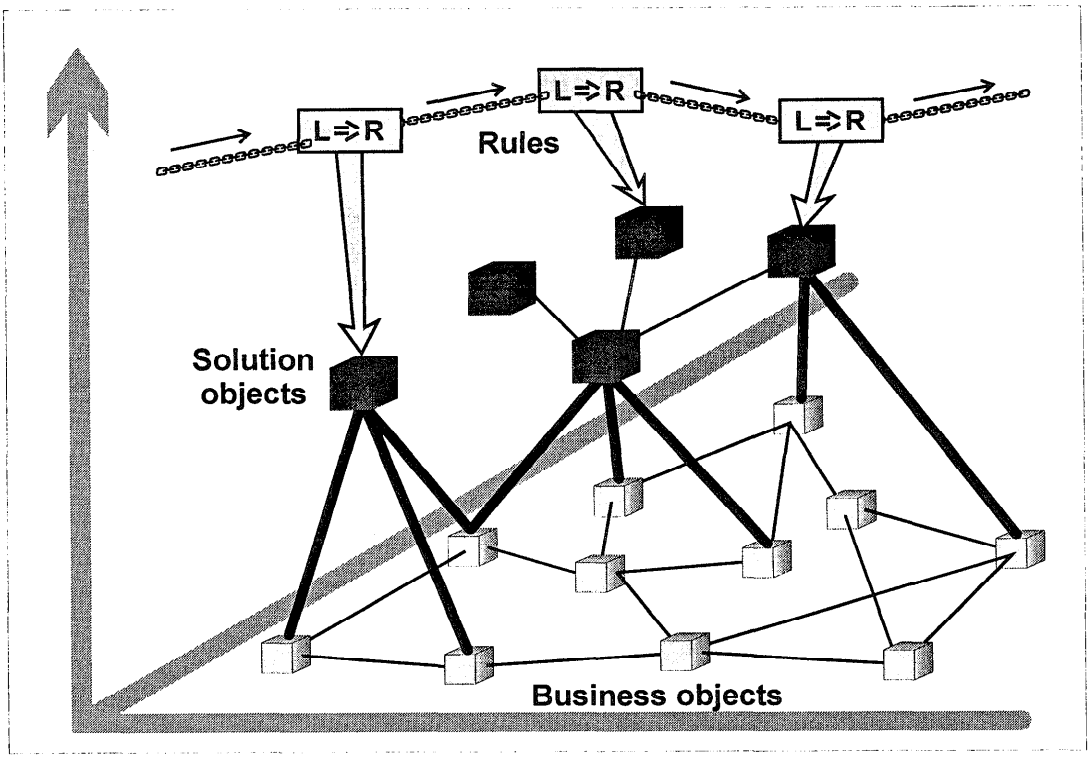

Figure 1. Three layered architecture

For example, a depot in the business is modelled as the 'depot' class, a depot's work shift is modelled 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 in Figure 3. The classes in the business object model provide services modelled 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 dependent on related properties of that 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 like 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 to 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 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. The processes are termed 'targets', 'best-of-type', 'assignment', and '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 over-commitment 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
3. Repeat 1. and 2. until there is either no space in the containers for the fillers or no fillers left.

This should be contrasted with naively putting fillers into containers without using the above. If we naively put fillers into containers simply on the basis of where the fillers would

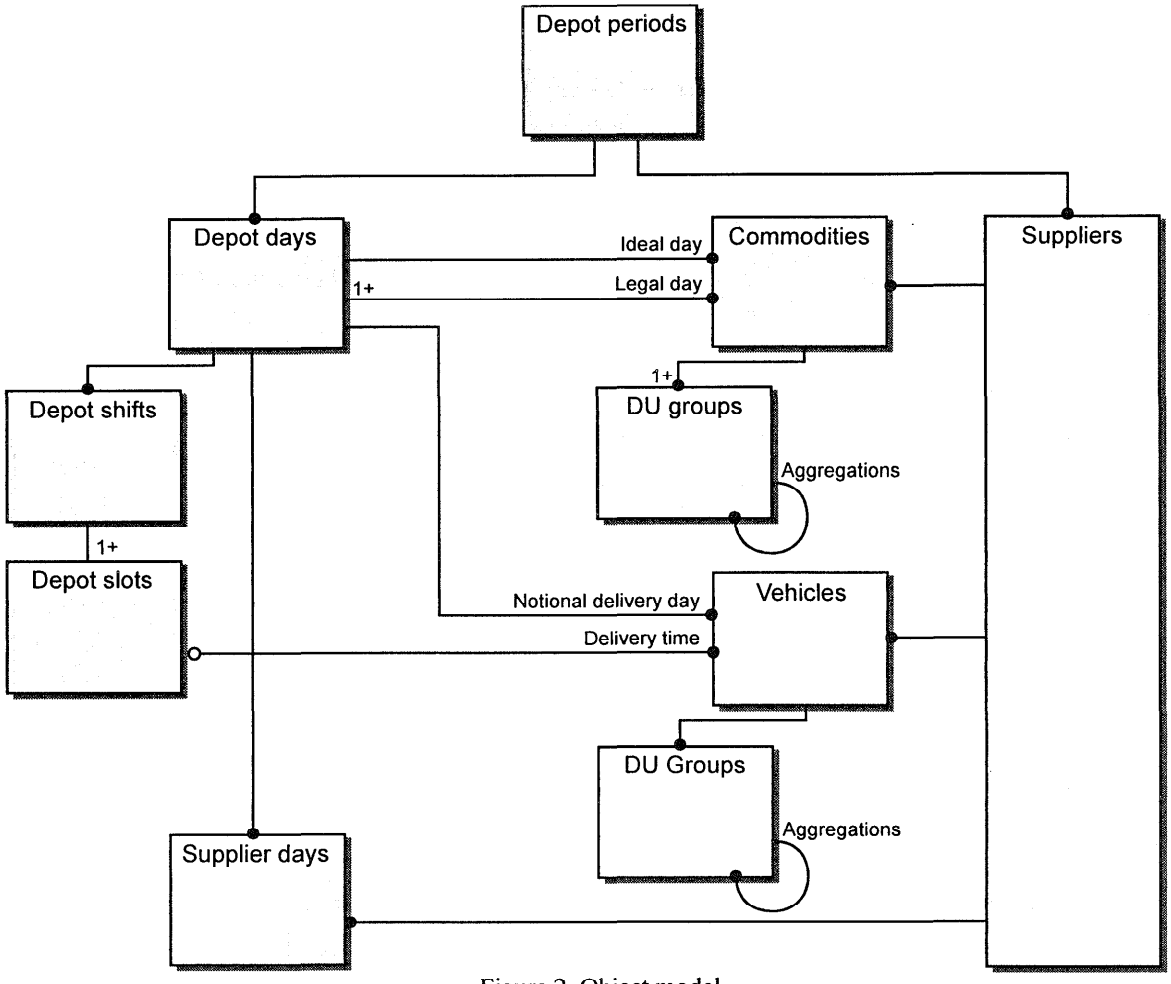

Figure 2. Object model

ideally like to go then we run out of capacity for the most over-committed containers and consume fillers that were permitted to go into less over-committed containers. This results in the most over-committed containers being full and the inability to fill the remaining containers as they are not permissible for the fillers remaining.

A supplier has a range of preferred times for delivery. 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 on 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 of the permissible linking relationships associated with the slot. This means that we can derive a value for the commitment for the receiving slot by aggregating deliveries 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 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 over-commitment for the slot or shift. The targets process sets up the infrastructure so that the least over-committed shift 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 of the same abstract type. Best-of-type provides meta-level information about the business object model. Best-of-type is employed where:

- The receiving slot is the parent and the preference links are the children

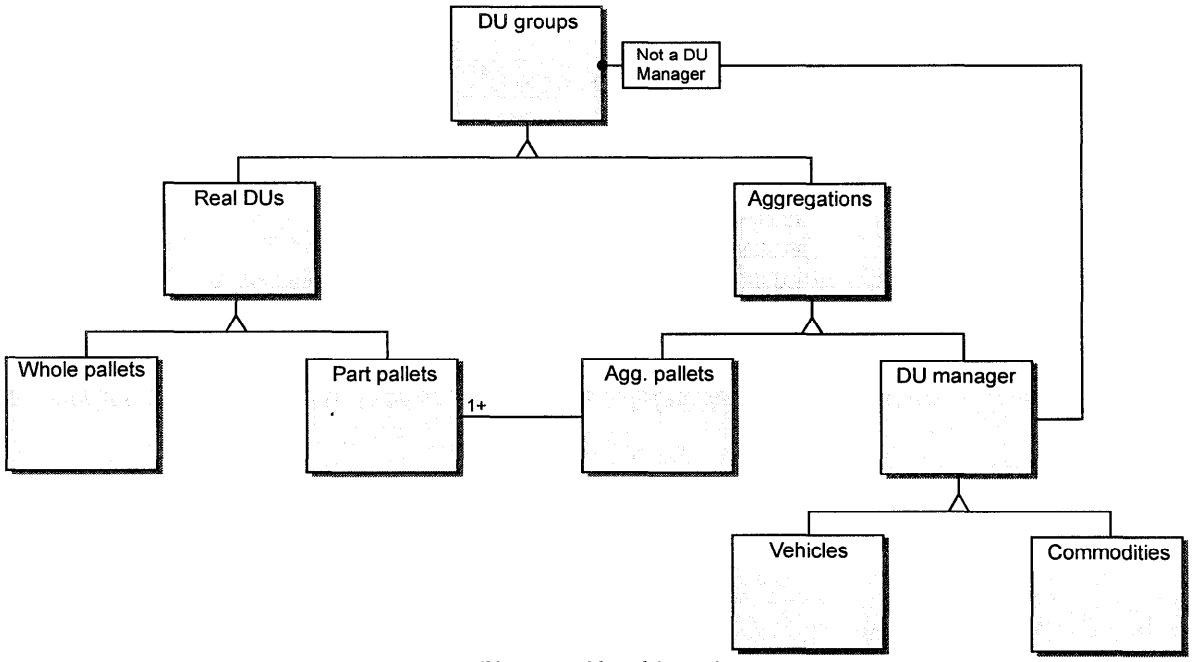

Figure 3. Class hierarchy

- Where the shift is the parent and where the slots are the children
- Where the depot day is the parent and where the shifts are the children.

The fitness function for the best preference link finds the best link by the following criteria: preference, supplier priority, commitment, vehicle size. The fitness function for the best slot uses the best-of-type for the link, over-commitment, and a weighting for assignments to each slot so far. The fitness function for the best shift uses the best slot and over-commitment 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 super-class where specific best-of-types are specialisations of the best-of-type class.

Assignment. Assignment is implemented as the rule 'focus-decision-demand-spreading'. The rete algorithm manages the rules 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 to 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 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. As the value of the assignment is debited from the slot so 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, over-commitment 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 composition of processes. To be able to do this, a mechanism is required that enables the process to traverse the relationships between the codependent 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' 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 specialised process classes

contain knowledge of their environment in terms of the map of classes and operations, and an intelligent traversal engine that allows them business process object to traverse the business object model whilst taking into account the state of the object model and not traversing 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 as to the nature of the assignment and traverse the object model ensuring that the receiving slots' and shifts' capacities are modified accordingly, that receiving slots and shifts are closed if there is no more capacity available; that 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 constraint processes, targets, best-of-type and assignment continues until either there is no capacity left on the slots and shifts for permissible assignments or there are no vehicles 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 via E.D.I. 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 interface to Ingres written in C.

SCION runs in a group of HP-UX UNIX machines, comprising 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 RDBMS 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 CPU mid-range machine with 512Mb RAM, optimised as a server; the 735's are smaller, single CPU machines with 80Mb RAM, optimised for processor speed. HP's TaskBroker job scheduling program is used to distribute jobs efficiently between clients and server and to 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. TaskBroker 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 be easily scalable for different data volumes.

Applications Innovation and Business Significance

The system can claim innovation in the following ways:

- Use of Artificial Intelligence techniques to automate time-constrained business processes
- Integration of rules-based, object orientation and relational paradigms whilst leveraging Artificial Intelligence approaches to provide a business solution
- It is an enabler for J. Sainsbury's business strategy
- It is mission critical to J. Sainsbury's business
- It is a key component in a reengineered business process
- The system's scale – the system processes up to 200,000 delivery units, producing 10,000 vehicles for 22 depots
- The system's performance – the system runs in a two-hour time window whilst resolving a complex task and processing large volumes of data
- The system's technical architecture exploits concurrency to perform its function.

Project History

The lifecycle of the project can be divided into three phases: the development cycle, the continuous improvement cycle and the maintenance cycle.

The development cycle took place between Spring of 1993 and Spring of 1994. An evolutionary model was adopted for the systems development. This process was managed using a time box approach for each stage, each stage producing a system deliverable. The key system deliverables were:

- Conceptual Demonstrator: May 1993 – Jun 1993
- Prototype: Jun 1993 – Feb 1994
- Production Prototype: Mar 1994 – Jun 1994
- Production System: Jul 1994 – Nov 1994.

Each deliverable was seen as a stage in the evolution toward a solution that met all the business' requirements. The primary drivers for this approach were 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 a 'big bang' approach. At each stage J. Sainsbury Management had a tangible software deliverable that they could assess and to which they could offer feedback prior to moving forward.

The software development approach was for each stage cyclical. The software development cycles comprised:

- Domain Knowledge Acquisition
- Business Analysis
- Solution Design
- Software Logical and Physical Design
- Incremental Build
- Expert Verification

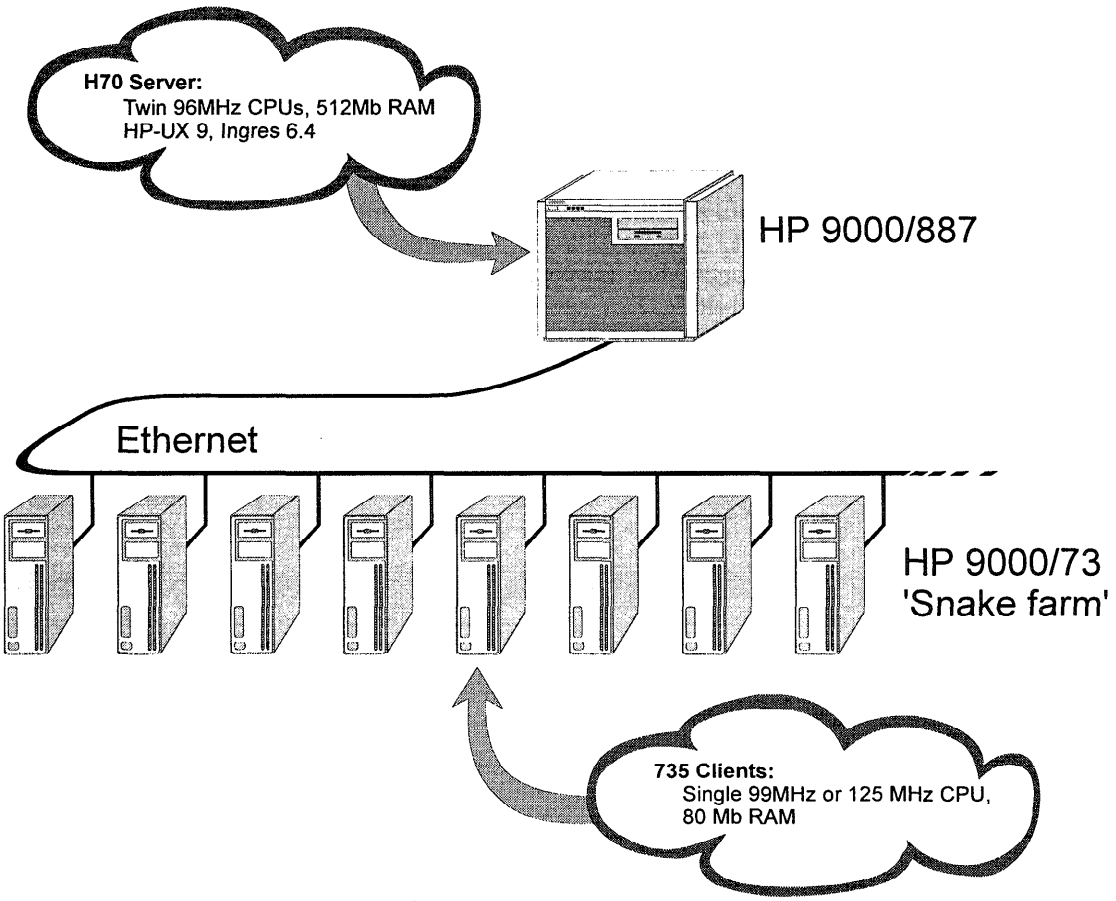

Figure 4. Hardware configuration

The evolutionary aspect of the lifecycle was implemented by significant design and code reuse between each stage of the lifecycle and the filtering out of approaches and mechanisms that were inefficient, non-robust 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 organisational and process levels in J. Sainsbury. This manifested itself in requirements for enhancements, tuning of the quality of existing functionality, and a drive to reduce the system's run time. During this period, the run time was halved from two hours to one hour.

The support phase of the system commenced in January 1995 and will be ongoing for the life of the system. A two-tier model was adopted for the support of the system. This consists 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 via SQL to remove the errant data that has caused the crash. The system is then restarted and run through to completion. Primary support requires no knowledge of the systems internal design or coding. An error recovery document contains the necessary knowledge and processes to resolve primary support problems. This document is supported by bespoke diagnostic tools.

Secondary support comes into play if a system crash cannot be resolved by manipulating the data. This entails accessing the system at the code level. This 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 very robust. The gearing of the development and implementation approaches have been such to ensure robustness. Evidence of this is that the SCION Depot Bookings system dealt with a 50% increase in data volumes over Christmas 1995 and ran within the operational time window. Nevertheless, due to 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 specialised process class from the abstract super-class. As the business object model represents business reality, the business classes can evolve without jeopardising the internal structural coherence of the system.

System Validation

The functional requirements of the application were validated through three distinct processes: Firstly using the J. Sainsbury Business Experts within the project team; secondly by user acceptance testing by the business; thirdly by feedback from the incremental rollout of the system.

The first process consisted of two months of business rules' 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 had been 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 a depot at a time until confidence was such that large numbers of depots 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 output enabled the business to judge the speed of rollout of the system. This approach enabled the business to derive business benefit whilst simultaneously building confidence in the results of the system.

The performance requirement of the system was validated by two benchmarking exercises. These 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. This consisted of taking production data and randomising the data variables in their respective valid ranges. This process is ongoing as it periodically yields data conditions that throw the system. These are trapped and their resolution incorporated in the system.

Application Deployment and Use

The SCION project has been implemented in two phases: The automation of Depot Bookings under a legacy weekly ordering system, and then to migrate to 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 SCION Depot Bookings per calendar month.

The second phase is ongoing and represents a radical change to existing operating procedures and processes. There are currently six depots running under the daily ordering regime.

Application Payoff

System Benefits

The system is a strategic enabler. As such, the primary benefits of the system are realised 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 are engaged in. Nevertheless, it is projected that the SCION project, comprising of SCION Ordering and Bookings, will produce benefits of more than £10 million in the next five years and return on investment in six 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 the result of SCION over the last year support these projections.

The other key benefits of the system are:

- Reduction in the amount of administration required to manage depot bookings, both at Head Office and for J Sainsbury's suppliers
- The Bookings system improves the utilisation of depot receiving resources
- It provides enhanced maintenance facilities for managing depot receiving capacities
- It supports new concepts critical to the reengineering of the supply chain
- It provides the business with control over the contents on vehicles hence supporting the management of transport costs.

Summary

The SCION Depot Bookings system automates the planning and scheduling of perishable and non-perishable commodities and the vehicles that carry them into J. Sainsbury depots. This is a strategic initiative, enabling the business to move to daily ordering. The system is mission critical, managing the inwards flow of commodities from suppliers into J. Sainsbury's depots. The system provides J. Sainsbury with control over the vehicles and goods coming into their 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 UK.

References

Alexander C. (1979). *The Timeless Way of Building.* Oxford: University Press.

Hammer M. and Champy J. (1993). *Re-engineering the Corporation: A Manifesto for the Business Revolution.* NY HarperCollins.

Hart A. (1989). *Knowledge Acquisition for Expert Systems.* Reading, MA: Addison-Wesley

Jacobs S. (1992). What is business process automation? *Expert Systems Applications* August, 5-10.

Rumbough J., Blaha M., Premerlani W. *et al.* (1991). *Object-oriented Modelling and Design.* Englewood Cliffs, NJ: Prentice-Hall.

Winston P.H., (1992). *Artificial Intelligence.* Reading, MA: Addison-Wesley.