

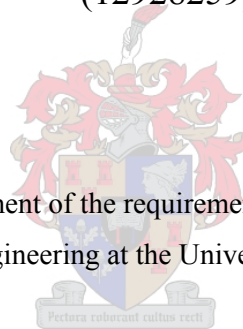
THE DEVELOPMENT OF A GENERIC JUST-IN-TIME SUPPLY CHAIN OPTIMISATION SOFTWARE TOOL

Presented by:

F.v.B. Bredenkamp

(12928259)

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in
Industrial Engineering at the University of Stellenbosch.



Study Leaders:

Prof. W. Van Wijck & Mr. J. Bekker

April 2005

DECLARATION

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: _____

Date: _____



ACKNOWLEDGEMENTS

My parents for always being there for me.

Lynne for her support and understanding.

Prof. Willie van Wijck for his guidance during this project.

James Bekker for taking over the reins as promoter at the most crucial time during this project.

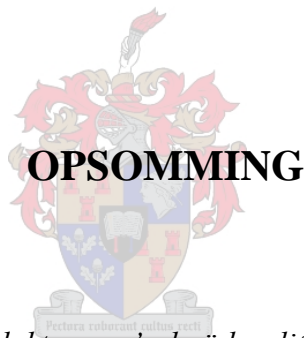
Izak Krige from SCM Systems (Pty) Ltd for financing the required software and his constant support throughout the project.

Dr. Deon Kellermann at Vynide for allowing me to finish my studies, by giving me time off when required.



ABSTRACT

The demand from modern day customers for quality products, supplied in any quantity and within a short lead-time, forces organisations to stock the correct amount of inventory in the correct locations in its supply chain. Establishing the correct inventory levels within an organisation's supply chain is complicated by the various stochastic processes occurring in a supply chain. The thesis is aimed at the development of a generic Just-In-Time (JIT) supply chain optimisation software tool, whereby the correct inventory levels for an organisation can be determined. These inventory levels will ensure that the organisation will achieve a predefined customer service level at the minimum cost to the company. The tool was developed and satisfactory results were obtained using the Harmony Search Algorithm (HSA) for optimising the inventory levels.



OPSOMMING

Hedendaagse kliënte verwag produkte van 'n hoë kwaliteit, verskaf in enige hoeveelheid en binne 'n kort leityd. Dit plaas geweldige druk op ondernemings om te verseker dat die korrekte hoeveelheid voorraad, in die korrekte plekke in hul voorsieningsketting geberg word. Die bepaling van die korrekte voorraadvlakke in 'n onderneming word verder gekompliseer deur die verskeie stogastiese prosesse wat binne 'n voorsieningsketting teenwoordig is. Die tesis is daarop gemik om 'n generiese knap-betydse voorsieningsketting optimerings programmatuur te ontwikkel. Die optimeringspakket bepaal die korrekte voorraad vlakke vir 'n organisasie, met die doel om 'n vooraf-bepaalde dienspeil te handhaaf teen die minimum koste. Die pakket het bevredigende resultate gelewer, met die "Harmony Search" Algorithme (HSA) wat vir die optimeringsproses gebruik is.

TABLE OF CONTENTS

LIST OF FIGURES.....	iv
LIST OF TABLES.....	vi
GLOSSARY	vii
CHAPTER 1 INTRODUCTION.....	1
CHAPTER 2 CONCEPT DESIGN.....	3
CHAPTER 3 THE OBJECTIVE FUNCTION.....	6
3.1 JUST-IN-TIME METHODOLOGY	7
3.2 FACTORS DETERMINING THE OPTIMUM KANBAN INVENTORY SOLUTION	8
3.2.1 Customer Service Level	8
3.2.2 Carrying Cost of Inventory.....	8
3.2.3 Acquisition Cost of Inventory	10
3.3 THE DEVELOPMENT OF THE INVENTORY OBJECTIVE FUNCTION	12
3.3.1 Constraint in Inventory Objective Function	12
3.3.2. Costs Fixed per Unit.....	13
3.3.3 Costs Fixed per Order.....	16
3.3.4 The Objective Function	17
CHAPTER 4 THE MODEL BUILDER.....	18
4.1 THE BOUNDARIES OF THE INVENTORY OPTIMISATION TOOL	18
4.2 DATA REQUIRED BY THE MODEL BUILDER	20
4.3 DESIGN OF THE DATABASE REQUIRED TO STORE THE MODEL BUILDER'S DATA	22
CHAPTER 5 THE SIMULATION MODEL.....	23
5.1 THE USE OF SIMULATION IN SUPPLY CHAIN OPTIMISATION.....	24
5.2 THE SUPPLY CHAIN CONCEPT MODEL	26
5.3 PARAMETERS TO BE INVESTIGATED	35
5.4 TRANSLATION OF CONCEPT MODEL TO COMPUTER MODEL	36

5.4.1 Simulation Software Package Used.....	37
5.4.2 Translation to a Generic Computer Model.....	38
5.5 ANALYSIS OF SIMULATION OUTPUT.....	47
5.5.1 Initial Simulation Run	47
5.5.2 Statistical Analysis of Output Parameters	49
5.6 SIMULATION MODEL'S DATABASE REQUIREMENTS	51
CHAPTER 6 THE OPTIMISATION ALGORITHM.....	54
6.1 THE SELECTION OF THE OPTIMISATION ALGORITHM	54
6.2 THE HARMONY SEARCH ALGORITHM.....	55
6.2.1 Parameters of the Harmony Search Algorithm	57
6.2.2 Convergence of the Harmony Search Algorithm	58
6.3 THE IMPLEMENTATION OF THE HARMONY SEARCH ALGORITHM	59
CHAPTER 7 VERIFICATION & VALIDATION	63
7.1 VERIFICATION OF THE OPTIMISATION TOOL	63
7.1.1 Model Builder.....	63
7.1.2 Simulation Model	64
7.1.3 Optimisation Algorithm.....	69
7.1.4 Report Generator	69
7.2 VALIDATION OF THE OPTIMISATION TOOL	70
CHAPTER 8 CONCLUSION & RECOMMENDATIONS	73
REFERENCES	75
BIBLIOGRAPHY.....	77
APPENDIX A THE USER'S GUIDE TO THE OPTIMISATION TOOL.....	A-1
APPENDIX B THE DATABASE'S DATA DICTIONARY	B-1
APPENDIX C THE SUPPLY CHAIN MODELS USED TO VERIFY THE WORKING OF THE SIMULATION MODEL	C-1
APPENDIX D THE SPREADSHEET USED IN VERIFYING THE SIMULATION MODEL	D-1
APPENDIX E THE SUPPLY CHAIN MODEL USED IN VALIDATING THE SUPPLY CHAIN OPTIMISATION TOOL	E-1

LIST OF FIGURES

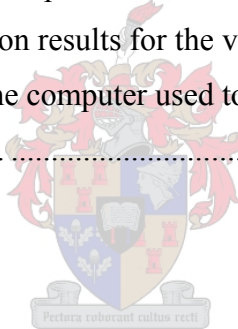
Figure 2.1: The various building blocks of the supply chain optimisation software tool.	3
Figure 2.2: A high level flow diagram showing the working of the inventory optimisation tool.....	4
Figure 3.1: The fluctuation of inventory levels in the economic order quantity inventory control model.....	10
Figure 3.2: The fluctuation of inventory levels in the Just-In-Time inventory control system.	10
Figure 3.3: The acquisition cost incurred for various order quantities when a replenishment order is placed.....	11
Figure 3.4: A flow diagram of how a possible inventory solution is evaluated using the inventory objective function.....	13
Figure 3.5: Diagram showing the relationship between the various costs.	16
Figure 4.1: A schematic diagram of the supply chain modelled by the optimisation tool.....	19
Figure 4.2: The sequence in which data is collected by the model builder.....	21
Figure 4.3: The entity relationship diagram of the database for modelling an organisation's supply chain.....	22
Figure 5.1: The handling of customer orders placed at a consignment warehouse.....	27
Figure 5.2: The handling of customer orders and stock replenishment orders placed at a regional warehouse.	28
Figure 5.3: The handling of customer orders and stock replenishment orders placed at a plant warehouse.	29
Figure 5.4: The handling of production orders placed at a plant.	30
Figure 5.5: The ordering and receiving of raw material replenishment orders at a plant's raw material warehouse.	31
Figure 5.6: The process followed at a plant warehouse when a production order is booked to the warehouse.	32
Figure 5.7: The process followed at a regional warehouse when a stock replenishment order is received from an upstream location in the supply chain.	33
Figure 5.8: The process followed at a consignment warehouse when a stock replenishment order is received from an upstream location in the supply chain.	34

Figure 5.9: The translation of the simulation concept model to a Simul8 computer model.	40
Figure 5.10: Discarding the transient phase (warm-up period) and batching the remaining observations. [Bekker 2003, p. 53].....	48
Figure 5.11: The revised entity relationship diagram.....	52
Figure 6.1: The basic pseudocode for the Harmony Search Algorithm [Bekker 2004b, p.19].....	56
Figure 6.2: Typical convergences patterns of the HSA. [Bekker 2004b, p. 23]	59
Figure 6.3: The working of the HSA in the optimisation tool.	60
Figure 6.4: The final entity relationship diagram accommodating the HSA's Harmony Memory (HM).	61



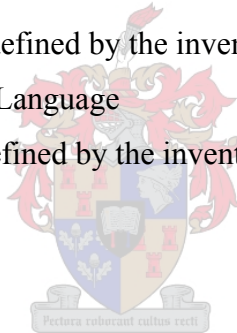
LIST OF TABLES

Table 4.1: The data fields stored in each of the database's tables.....	23
Table 5.1: The requirements of the simulation software package to model the generic supply chain.....	37
Table 5.2: Desired confidence half-width values (h^*) for the parameters under study.....	50
Table 5.3: The database's tables together with their respective fields.	53
Table 6.1: Comparison between optimisation and musical performance. [Geem et al. 2001, p. 62].....	56
Table 6.2: The tables and their respective fields, to accommodate the HSA's Harmony Memory (HM).	62
Table 7.1: The binary search procedures with their respective input variables.....	68
Table 7.2: Validation values for the optimisation tool.	71
Table 7.3: Comparison of validation results for the various scenarios.....	72
Table 7.4: The specifications of the computer used to verify and validate the supply chain optimisation tool.	72



GLOSSARY

BOM	Bill of Materials
CI	Confidence Interval
CSL	Customer Service Level
EOQ	Economic Order Quantity
HSA	Harmony Search Algorithm
HM	Harmony Memory.
HMCR	Harmony Memory Considering Rate
IOF	Inventory Objective Function
JIT	Just-In-Time
ODBC	Open Data Base Connection
PAR	Pitch Adjustment Rate
R	Reorder Cost, as defined by the inventory objective function in Chapter 3.
SQL	Structured Query Language
T	Transport Cost, defined by the inventory objective function in Chapter 3.



CHAPTER 1

INTRODUCTION

The increasing demand from customers for quality products, supplied in any quantity and within a short lead-time forces organisations to stock enough of the correct inventory at the correct locations within its supply chain. Determining how much and of what inventory to stock in the various locations in a supply chain is not a trivial task, since a supply chain is filled with various stochastic processes. Each of these processes has a direct influence on what inventory, and how much of it, is stored in each of the locations in a supply chain.

If an organisation is to stay competitive and ahead of its competitors it must adhere to the market's demand while keeping its costs low, where the costs associated with inventory can be quite significant. The most apparent inventory costs are the capital interest lost due to capital tied up in inventory together with the holding and transportation cost associated with inventory. A second and unquantifiable cost directly related to inventory is the cost of not being able to meet a customer's demand, since the true cost of not making the sale can be much higher than the actual sale lost.

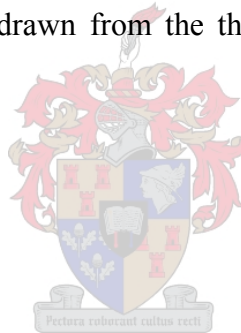
SCM Systems (Pty) Ltd is a South African company which offers support services to other logistic companies in South Africa. This thesis is as a result of one of the projects undertaken by SCM Systems for one of their clients. The purpose of the thesis is to develop a generic supply chain optimisation tool by which an organisation using the Just-In-Time (JIT) methodology to manage their supply chain, can determine the optimum inventory levels throughout its supply chain. The optimum inventory levels will thus ensure that a predefined customer service level is maintained at the minimum cost to an organisation.

The thesis is structured as follows: Chapter 2 gives a brief overview of the various concepts from which the optimisation tool is compiled together with an overview of the high level working of the tool. This is followed by a discussion on the development of the objective function used in the optimisation process in Chapter 3.

Chapters 4 to 6 are each concerned with the detailed design of the concepts discussed in Chapter 2, starting with a discussion on the development of the model builder in Chapter 4. The model builder acts as the user interface which is responsible for gathering the required data to model an organisation's supply chain effectively via the tool.

Chapter 5 is concerned with the development of the simulation model used by the optimisation tool in order to model the various stochastic processes occurring in a supply chain. This is followed by a discussion on the implementation of the optimisation algorithm within the optimisation tool in Chapter 6. The optimisation algorithm is the driver of the whole optimisation process and is ultimately responsible for the inventory solution proposed by the tool.

The thesis is concluded in Chapters 7 and 8, where Chapter 7 is concerned with the processes followed in the verification and validation of the optimisation tool. Chapter 8 presents the various conclusions that can be drawn from the thesis together with recommendations for future fields of study.



CHAPTER 2

CONCEPT DESIGN

The development of the supply chain optimisation software tool, as described in Chapter 1, combines various concepts to form the final deliverable. This chapter describes these concepts from which the optimisation tool is compiled, and gives a broad overview of the functioning of the tool.

The optimisation tool consists of five main building blocks: the model builder, the database, the simulation model, the optimisation algorithm and the report generator. These building blocks and the interactions between them are shown schematically in figure 2.1, where the sequence in which the mentioned building blocks interact with each other is dictated by the working of the optimisation tool.

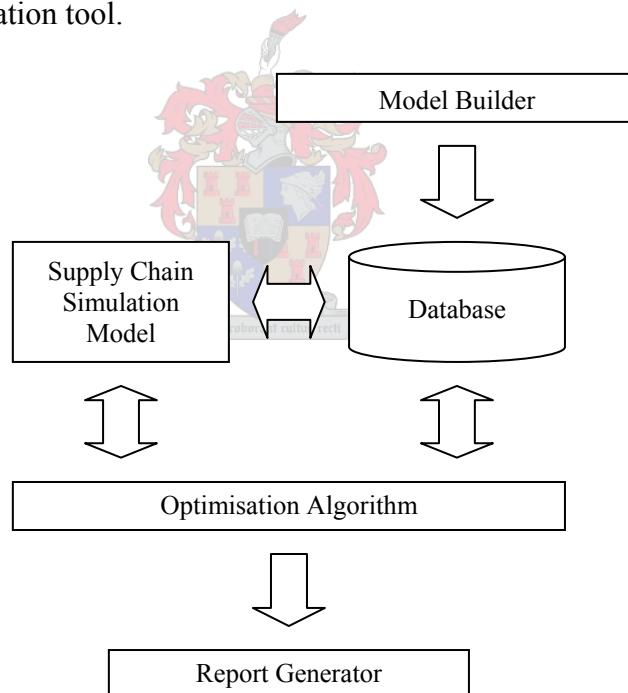


Figure 2.1: The various building blocks of the supply chain optimisation software tool.

A high level program flow diagram showing the working of the optimisation tool is shown in figure 2.2.

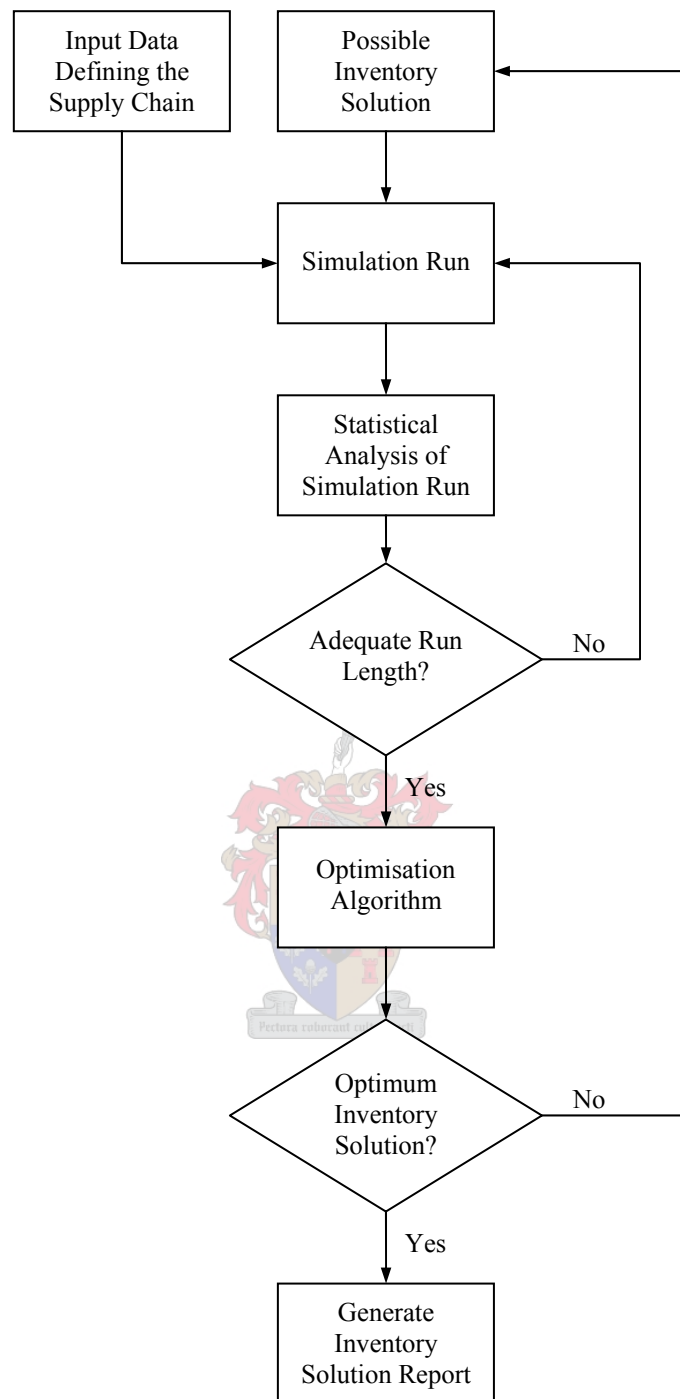


Figure 2.2: A high level flow diagram showing the working of the inventory optimisation tool.

The model builder, shown in figure 2.1, is responsible for gathering all the required data to model an organisation's supply chain effectively. The model builder can be seen as the user

interface, prompting the user to fill in various fields as the user defines his supply chain within the optimisation tool. The data gathered by the model builder is stored in the database. The database is, however, not just responsible for storing the data gathered by the model builder, but also acts as the medium used by the simulation model and the optimisation algorithm to pass data between each other before and after each simulation run.

The simulation model, as shown in figure 2.1, is built using a simulation software package and models an organisation's supply chain according to the data stored in the database. After each simulation run statistical analysis is done on each of the simulation model's output parameters. This is done to ensure that each output parameter has a small enough confidence interval. If the confidence interval of one or more of the output parameters are not small enough, the run length is extended to ensure a sufficient confidence interval; otherwise the optimisation algorithm is notified of the completed run by the simulation model directly, and the output of the simulation run is passed-on to the optimisation algorithm via the database.

The optimisation algorithm analyses the output of each simulation run and determines if the proposed inventory solution is optimal. If this is not the case, a new solution is created, based on previous proposed solutions together with their respective simulation results, and is then sent to the simulation model via the database. Conversely, if the proposed solution is optimal the report generator compiles a report, presenting the various stock levels of each product for each location defined within the supply chain.

In order for the optimisation algorithm to evaluate a proposed inventory solution, an objective function is implemented in the simulation model. The objective function is defined in terms of various variables in the simulation model and relates these values to a single output variable for each simulation run. The optimisation algorithm adapts the proposed inventory solution in an attempt to minimize the output of the objective function.

It is important that the objective function relates the output of the simulation model as accurately as possible to real-life, since an inaccurate objective function will lead to a sub-optimal inventory solution. Given that the objective function plays such an important role in determining the optimal inventory solution, Chapter 3 is devoted entirely to the choice and justification of the objective function implemented in the simulation model. This concludes the discussion of the optimisation tool's concept design.

CHAPTER 3

THE OBJECTIVE FUNCTION

In order to optimise the inventory levels throughout an organisation's supply chain, the optimisation algorithm requires an objective function which is implemented within the simulation model. After each run of the simulation model the objective function returns a value to the optimisation algorithm. The objective function's value is in the long run minimised by the algorithm in order to minimise the stock levels within the simulation model. The development of the objective function is discussed in this chapter.

The primary aggregate performance measure of inventory management within an organisation's supply chain is based on the size of the inventory investment within the supply chain. Inventory turnover and weeks of supply are two measures of the inventory investment relative to the total cost of goods that are provided through the supply chain [Silver et al. 1998, p. 16]. These are:

$$\text{Inventory turnover} = \frac{\text{Annual cost of goods sold}}{\text{Average aggregate inventory value}} \quad \dots(3.1)$$

$$\text{Weeks of supply} = \left(\frac{\text{Average aggregate inventory value}}{\text{Annual cost of goods sold}} \right) 52 \text{ weeks} \quad \dots(3.2)$$

An equally important measurement in an organisation is its customer service level, where an organisation's customer service strategy is the driving force behind the design and operation of its supply chain. [Shapiro 2001, p. 287] "Management is concerned with overall inventory levels and customer service levels (by broad classes of items) as opposed to, for example, minimization of costs on an individual item basis." [Silver et al. 1998, p. 24] The purpose of inventory within an organisation is thus to ensure that a certain customer service level is maintained by the organisation. Therefore customer service level dictates the manner in which an organisation structures and operates its supply chain, as well as the inventory within its supply chain. This also implies that the cost associated with the manner in which an organisation structures and operates its supply chain is due to the organisation's customer

service strategy. The Just-In-Time methodology, applied to inventory management, is discussed in the next section.

3.1 JUST-IN-TIME METHODOLOGY

The Just-In-Time (JIT) manufacturing methodology is aimed at achieving high levels of production throughput using minimal inventories of raw materials, work-in-process and finished goods. The same methodology can be applied to a supply chain where the aim is achieving a high level of customer service using minimal inventories of raw materials, work-in-process and finished goods [Dobler & Burt 1996, p. 533].

This is achieved by assigning a kanban size and a number of kanbans to each product and raw material for each of the locations within an organisation's supply chain. The word kanban means "sign" or "instruction card" in Japanese. In a paperless inventory control system, containers are used instead of instruction cards. Thus kanban size implies the number of products or raw materials that a card represents, or that is stored in a container. The number of kanbans merely states the number of cards or containers required for each product or raw material. The demand for a product at a location in the supply chain influences the kanban size and the number of kanbans of the product stored at that specific location. This implies that the kanban sizes and number of kanbans, of a specific product or raw material, will differ among the various storage locations within an organisation's supply chain.

The relationship between a product's kanban size and kanban number is inversely proportional for a given demand, lead time and a small quantity of safety stock [Chase et al. 2001, p. 401]:

$$\text{Number of Kanbans} = \frac{\text{Average Demand during lead time} + \text{Safety Stock}}{\text{Kanban Size}} \quad \dots(3.3)$$

As mentioned above, the customer service strategy governs the way in which an organisation structures and operates its supply chain. In structuring an organisation's supply chain according to the Just-In-Time methodology, a company is attempting to maintain a predefined customer service level with the least amount of capital investment in inventory. The factors

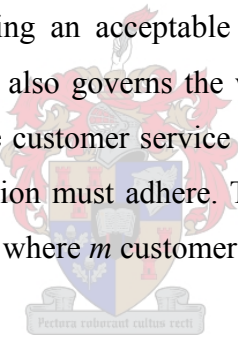
that determine the optimum kanban inventory solution, for an organisation as a whole, is discussed in the following section.

3.2 FACTORS DETERMINING THE OPTIMUM KANBAN INVENTORY SOLUTION

Within any organisation inventory influences mainly three key areas of business: customer service level, carrying cost of inventory and acquisition cost of inventory. The optimum kanban inventory solution will be determined according to the way in which these three areas are affected by the solution. In the following paragraphs these three areas are discussed in detail regarding how the Just-In-Time methodology influences each of them.

3.2.1 Customer Service Level

As mentioned previously, achieving an acceptable customer service level is not only the purpose of holding inventory, but also governs the way in which an organisation structures and supports its supply chain. The customer service level can thus be seen as a constraint to which any kanban inventory solution must adhere. The author chose to define the customer service level (CSL), for a period P where m customer orders are placed, as follows:



$$CSL(P) = \frac{\sum_{n=1}^m SQ_n}{\sum_{n=1}^m OQ_n} \quad \dots(3.4)$$

Where:

OQ_n – Order quantity placed by customer order n .

SQ_n – Quantity of customer order n immediately dispatched.

m – Number of customer orders received in period P .

3.2.2 Carrying Cost of Inventory

The carrying cost of inventory consists of five elements: opportunity cost of the inventory investment, insurance cost, property tax, storage cost and obsolescence and deterioration cost.

[Dobler & Burt 1996, p. 523] In the Just-In-Time (JIT) methodology the carrying cost of inventory is not as significant as in traditional inventory control methodologies, since the aim of the JIT methodology is to limit the capital investment in stock, while maintaining a predefined customer service level. In the JIT methodology the size of the kanbans and the number of kanbans of each product and raw material within a supply chain determine the total capital investment in inventory. If one is to take an organisation's whole supply chain into account, i.e. not just focusing in on a specific location within the supply chain, the capital invested in inventory maintains an almost constant value over time, when applying the JIT methodology. This is due to the fact that the JIT methodology is far more geared towards the stabilisation of the inventory levels throughout the supply chain than the traditional fixed-order quantity methodology, also known as the economic order quantity model (EOQ). The working of the EOQ model is shown in figure 3.1, where a replenishment order of quantity Q is placed the moment the inventory reaches a level R . [Chase et al. 2001, p. 517] The virtual stock level shown in figure 3.1 consists of the sum of the on-hand inventory of a product currently stored at a location, and the inventory that is *en route* to that specific location.

In comparing the virtual stock level of the EOQ model to that of the JIT model, shown in figure 3.2, it can be seen that the JIT model's virtual stock level fluctuates far less than that of the EOQ model. It should also be noted that the average stock level of a product in the JIT methodology is very close to the maximum number of units stored, compared to the EOQ model where the average stock level is almost half of the maximum number of units stored. This is due to replenishment orders being placed for smaller quantities and more regularly in the JIT model, than orders placed in the EOQ model. As a result of a more frequent placement of replenishment orders a great deal of an organisation's inventory is in transit between the various locations within its supply chain.

All five components comprising the carrying cost of inventory can be calculated on a per unit per time basis. There are however two distinct groups into which the carrying cost can be divided, namely: cost per unit per time and cost per cubic meter per time. The opportunity cost of the inventory investment, insurance cost and obsolescence and deterioration cost of inventory all form part of the cost per unit per time group, while property tax and the storage cost of inventory falls into the cost per cubic meter per time group.

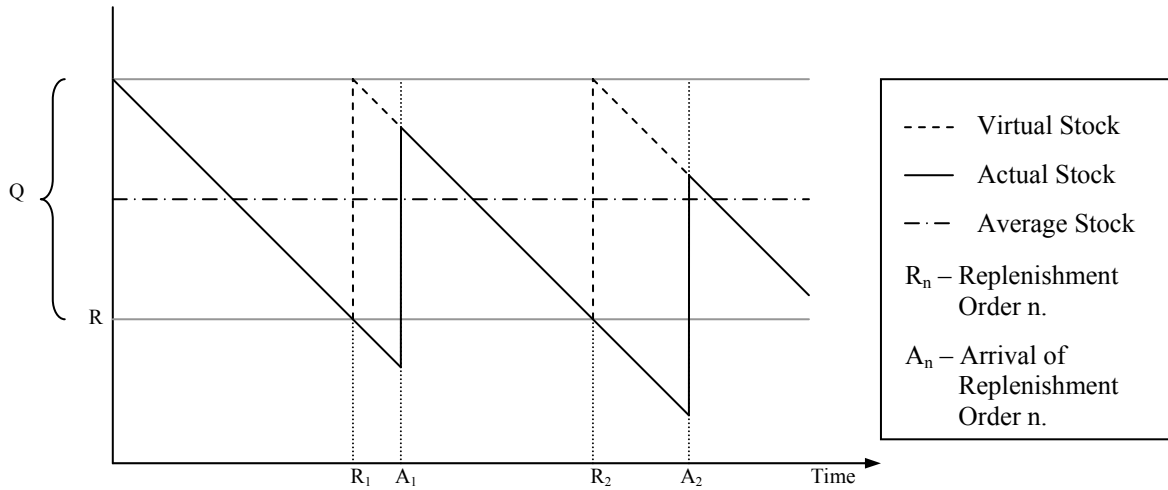


Figure 3.1: The fluctuation of inventory levels in the economic order quantity inventory control model.

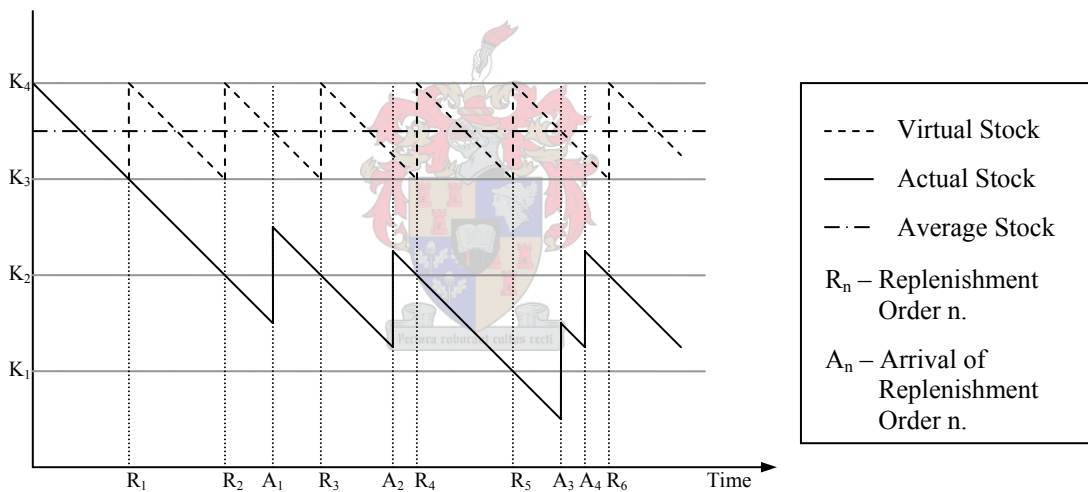


Figure 3.2: The fluctuation of inventory levels in the Just-In-Time inventory control system.

3.2.3 Acquisition Cost of Inventory

In contrast to the carrying cost, the acquisition cost is the indirect cost associated with the generating, handling and processing of an order. [Dobler & Burt 1996, p. 525] It consists of a portion of the wages and operating expenses, the cost of supplies and the cost of services. The acquisition cost associated with JIT can be quite considerable, because due to the low capital investment in inventory, the transportation of products and raw materials between the various locations within an organisation’s supply chain is extensive.

The choice of kanban size and the number of kanbans can influence the acquisition cost of inventory, while maintaining the same capital investment in inventory, for example: Product A can consist of 4 kanbans with a size of 4 each, the total number of items in stock is thus 16. It is however possible to group Product A into 2 kanbans with a size of 8 each. The total number of items in stock is thus still 16, but replenishment orders will be placed less frequent than in the first scenario, which will definitely decrease the acquisition cost of inventory.

Just as with the carrying cost of inventory, the acquisition cost of inventory can also be divided into two groups, namely: costs fixed per unit and costs fixed per order. The costs fixed per unit are similar to the carrying cost, whereas the costs fixed per order are the costs incurred each time inventory is ordered, irrespective of the quantity ordered. Figure 3.3 shows the acquisition cost incurred for various order quantities when a replenishment order is placed.

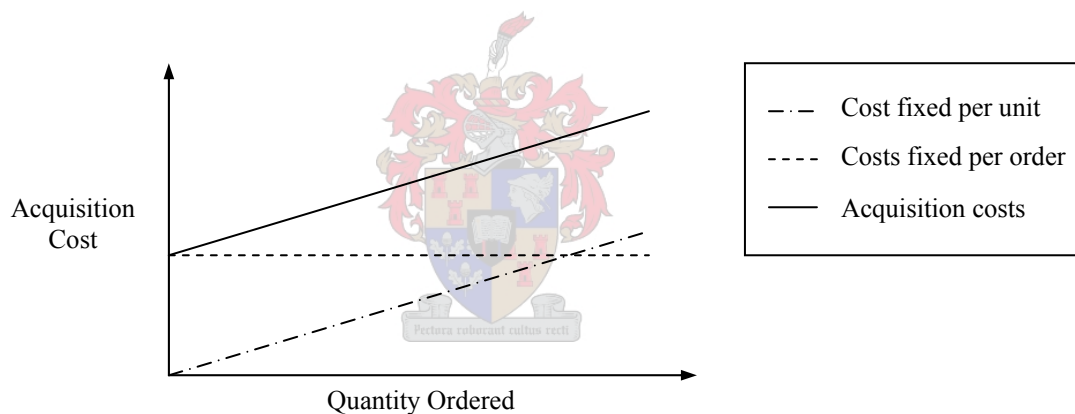


Figure 3.3: The acquisition cost incurred for various order quantities when a replenishment order is placed.

The three areas discussed in this section, namely: customer service level, carrying cost of inventory and the acquisition cost of inventory, are influenced by an organisation's inventory and will form part of the objective function used to determine the optimum kanban inventory solution. The development of the objective function is discussed in the following section.

3.3 THE DEVELOPMENT OF THE INVENTORY OBJECTIVE FUNCTION

The Just-In-Time methodology focuses on minimising the inventory levels throughout the supply chain, while maintaining a predefined customer service level. The kanban structure is the vehicle used within the JIT methodology to signal the placement of replenishment orders from upstream locations within the supply chain. The manner in which an organisation formulates its kanbans influences the organisation's customer service level, carrying cost of inventory and acquisition cost of inventory, as discussed in 3.2. To quantify these with a single value an objective function is required. The function is subsequently discussed.

3.3.1 Constraint in Inventory Objective Function

The purpose of the optimisation tool is to determine the optimum inventory solution, while maintaining a predefined customer service level. Thus any inventory solution must adhere to the predefined customer service level (CSL) in order to be considered as a possible solution. Figure 3.4 shows the flow diagram of the method used to evaluate a possible inventory solution when the inventory objective function is used. The pre-defined customer service level serves as a constraint to the inventory objective function, therefore equation 3.4 is re-written in equation 3.5, showing the constraint condition that the customer service level must adhere to:

$$\begin{aligned}
 CSL(P) &= \frac{\sum_{n=1}^m SQ_n}{\sum_{n=1}^m OQ_n} \\
 &\geq CSL_{Pre-defined}
 \end{aligned}
 \tag{3.5}$$

The inventory objective function is therefore made-up of the two remaining elements discussed in 3.2, namely: carrying and acquisition cost of inventory. Both of these costs can further be split into two groups: costs fixed per unit and costs fixed per order. It is these two costs out of which the objective function is ultimately formed. Both these costs are discussed in detail in the rest of this section.

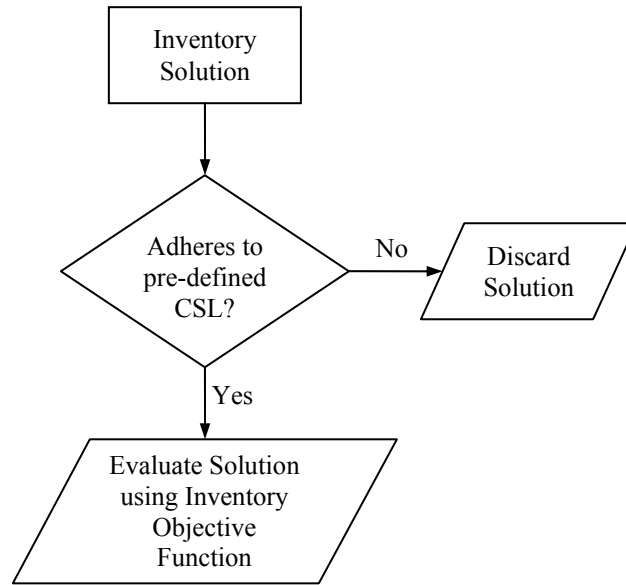


Figure 3.4: A flow diagram of how a possible inventory solution is evaluated using the inventory objective function.

3.3.2. Costs Fixed per Unit

The inventory cost assigned to a unit of inventory is made-up of all the inventory carrying costs plus a portion of the acquisition costs of inventory, as discussed in 3.2.2 and 3.2.3 on pages 8 and 10 respectively. The costs fixed per unit of inventory can further be divided into three groups, namely: costs per unit per time, costs per cubic metre per time and costs per unit.

3.3.2.1 Costs Fixed per Unit per Time

The costs fixed per unit per time are mainly the opportunity cost of the inventory investment, insurance cost and obsolescence and deterioration cost of inventory. The following three assumptions are made with regard to the above mentioned costs:

Assumption 1: Low inventory levels in the JIT environment leads to a high inventory turnover, thus the obsolescence and deterioration cost of inventory is negligibly small.

Assumption 2: It is possible to assign the insurance cost of inventory as a portion of the capital value of inventory. This implies that a unit of inventory will cost more due to insurance paid for that unit over a period P .

Assumption 3: The average capital investment in inventory for the period P is equal to the maximum inventory size, as shown in figure 3.2 on page 10. The maximum inventory size is purely the sum of all the products' kanban sizes, multiplied by their kanban numbers throughout the whole supply chain.

It is now possible to express the inventory costs fixed per unit for period P as the opportunity cost of the capital investment in inventory (OC) for the period P , as shown in equation 3.6.

$$OC(P) = I \sum_{i=1}^m \sum_{j=1}^{n_i} C_{ij} S_{ij} \quad \dots(3.6)$$

Where:

m – Number of storage locations in the supply chain.

n_i – Number of products stored at location i .

C_{ij} – Capital invested in one unit of product j stored at location i .

S_{ij} – Maximum number of units stored of product j at location i .

I – Capital interest rate for the period P .

In equation 3.6 the maximum number of units stored of any product (S_j), in any location within the supply chain, is purely the multiplication of the product's kanban size with its kanban number.

3.3.2.2 Costs Fixed per Cubic Meter per Time

Property tax and the storage cost of inventory are both carrying costs of inventory and can be calculated on a cost per cubic metre per time basis. The following two assumptions are made:

Assumption 1: Property tax and the storage cost of inventory can be added together to form one cost.

Assumption 2: The storage facility's required size will be dependant on the maximum inventory size, as determined by the defined kanbans for the various products stored in the facility. Thus the fact that part of the inventory will always be on-route to the facility is ignored.

The inventory costs fixed per cubic meter for the period P is shown in equation 3.7, where the inventory costs fixed per cubic meter is termed the storage cost of inventory (S) for the period P .

$$S(P) = P \sum_{i=1}^m \left(LC_i \sum_{j=1}^{n_i} S_{ij} D_j \right) \quad \dots(3.7)$$

Where:

P – Period over which the storage cost is determined, measured in days.

m – Number of storage locations in the supply chain.

n_i – Number of products stored at location i .

LC_i – Location i 's storage cost per cubic meter per day.

D_j – Cubic meter dimensions of one unit of product j .

S_{ij} – Maximum number of units stored of product j at location i .

3.3.2.3 Costs Fixed per Unit

The inventory costs fixed per unit consist of a portion of the acquisition cost of inventory. This is the cost incurred each time a unit of inventory is transported between the various locations in the supply chain. The cost fixed per unit over a period P is termed the transport unit cost of inventory (T) and is determined as follows:

$$T(P) = \sum_{i=1}^m \sum_{j=1}^{n_i} t_{ij} X_{ij} \quad \dots(3.8)$$

Where:

m – Number of storage locations in the supply chain.

n_i – Number of products stored at location i .

t_{ij} – The replenishment unit cost of product j at location i .

X_{ij} – The number of units replenished of product j at location i .

3.3.3 Costs Fixed per Order

Conversely to the inventory costs fixed per unit, the inventory costs fixed per order comprises only a portion of the acquisition cost of inventory. This is the cost incurred each time a stock replenishment order is placed and includes costs such as import duties, telephone calls, stock consolidator's fee, etc. The inventory cost fixed per order for a period P is termed the replenishment order cost of inventory (R), as shown in equation 3.9.

$$R(P) = \sum_{i=1}^m \sum_{j=1}^{n_i} r_{ij} Y_{ij} \quad \dots(3.9)$$

Where:

m – Number of storage locations in the supply chain.

n_i – Number of products stored at location i .

r_{ij} – The replenishment unit cost of product j at location i .

Y_{ij} – The number of replenishment orders placed for product j at location i .

The relationships between the various costs, as discussed in 3.3.2 and 3.3.3 are shown in figure 3.5.

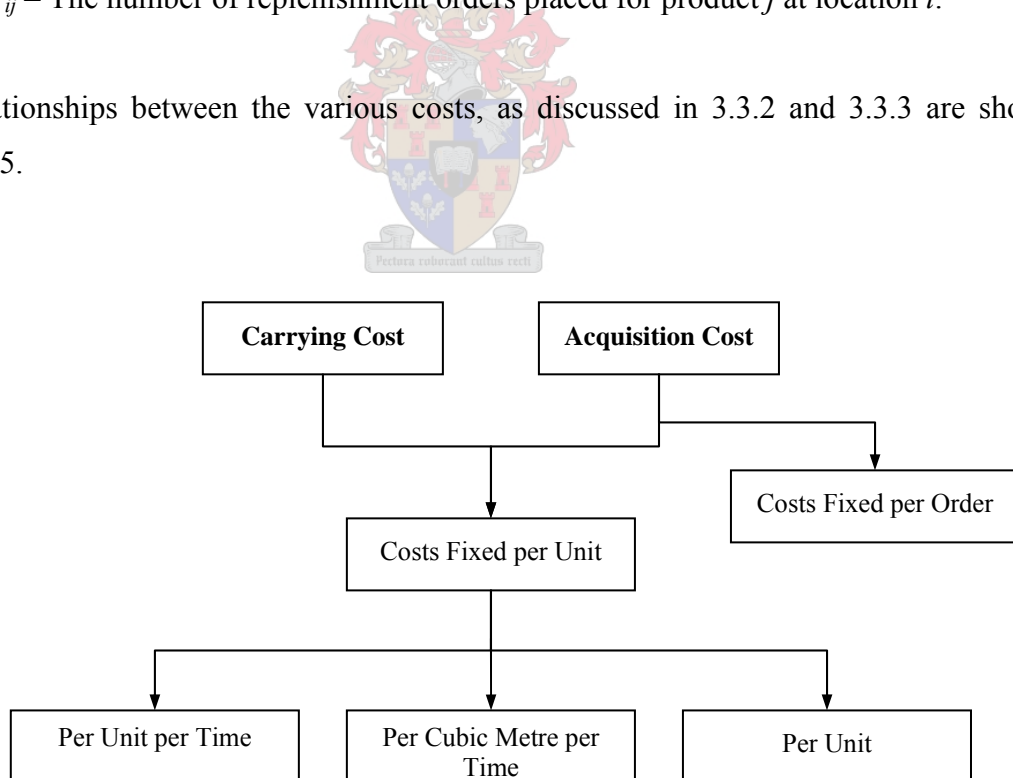


Figure 3.5: Diagram showing the relationship between the various costs.

3.3.4 The Objective Function

The inventory objective function (IOF) incorporates all of the above-mentioned costs, where the customer service level is a constraint to the objective function. The objective function is shown in equation 3.10:

$$\begin{aligned}
 IOF(P) &= \text{Costs Fixed per Unit} + \text{Costs Fixed per Order} \\
 &= \left(I \sum_{i=1}^m \sum_{j=1}^{n_i} C_j S_j \right) + P \sum_{i=1}^m \left(LC_i \sum_{j=1}^{n_i} S_j D_j \right) + T(P) + R(P)
 \end{aligned} \tag{3.10}$$

Subjected to :

$$\begin{aligned}
 CSL(P) &= \frac{\sum_{n=1}^m SQ_n}{\sum_{n=1}^m OQ_n} \\
 &\geq CSL_{Pre-defined}
 \end{aligned}$$

The inventory objective function, as shown in equation 3.10, is the thread that binds the next three chapters together: The objective function influences the data required to model the supply chain, discussed in Chapter 4. In Chapter 5 the components of the inventory objective function are implemented within the simulation model. Finally the minimisation of inventory levels, by the optimisation algorithm using the objective function, is discussed in Chapter 6.

CHAPTER 4

THE MODEL BUILDER

The supply chain inventory optimisation tool is comprised of various building blocks, as discussed in Chapter 2. These building blocks are: a model builder, database, simulation model, optimisation algorithm and a report generator. This chapter focuses on the detail design of the model builder and its influence on the design of the database. The design of the simulation model and optimisation algorithm, together with the report generator, is discussed in chapters 5 and 6 respectively.

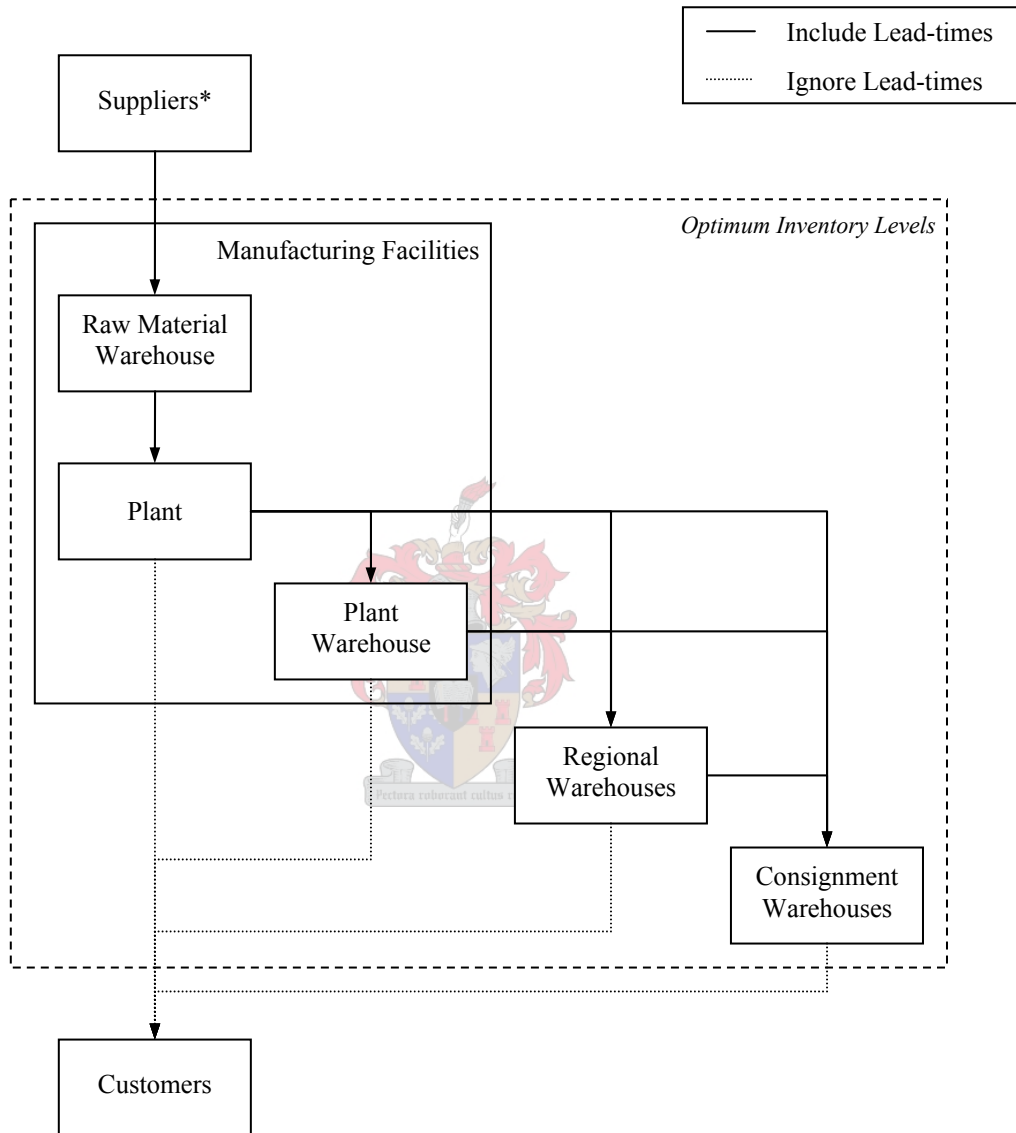
4.1 THE BOUNDARIES OF THE INVENTORY OPTIMISATION TOOL

If an organisation's optimum kanban inventory solution is to be determined, the inventory optimisation tool must model an organisation's supply chain within the simulation model. In the optimisation tool the model builder is responsible for gathering the data required to model the supply chain, where the data gathered by the model builder is stored in the database. The supply chain configuration catered for in the simulation model is shown in figure 4.1.

The supply chain configuration is fed by various suppliers of raw materials. The raw materials are stored in raw material warehouses, where each warehouse supplies directly to a manufacturing plant. The plant converts the raw materials into finished goods and stores them in its finished goods warehouse or, if it is a make-to-order product, sends it directly to the customer. From the finished goods warehouses, goods can be dispatched directly to customers, regional warehouses or consignment warehouses. Similarly goods in a regional warehouse can be dispatched directly to customers or to consignment warehouses, where a consignment warehouse supplies only to a specific customer.

The simulation model is triggered by customers placing orders for different products at various locations within the supply chain. The purpose of the model is to determine the optimum kanban inventory solution for each location within the supply chain, with the intention of ensuring a predefined customer service level for the organisation as a whole. Thus the in-transit lead-times from any location within the supply chain to customers are omitted in

the simulation model, since inventory in the supply chain has no influence on this specific lead-time. Conversely the raw material replenishment lead-times are built into the simulation model, because their influence on customer service level can be compensated for by inventory stored at locations in the supply chain.



* Not catering for out of stock scenario.

Figure 4.1: A schematic diagram of the supply chain modelled by the optimisation tool.

The lead-time of a replenishment order will vary for various reasons; this is modelled in the simulation model by supplying a standard deviation (normally distributed lead-times assumed) to the average replenishment lead-time of a raw material. The following assumption is made with regard to all suppliers:

Assumption: The suppliers of raw materials will never be out of stock when a replenishment order is placed in the simulation model.

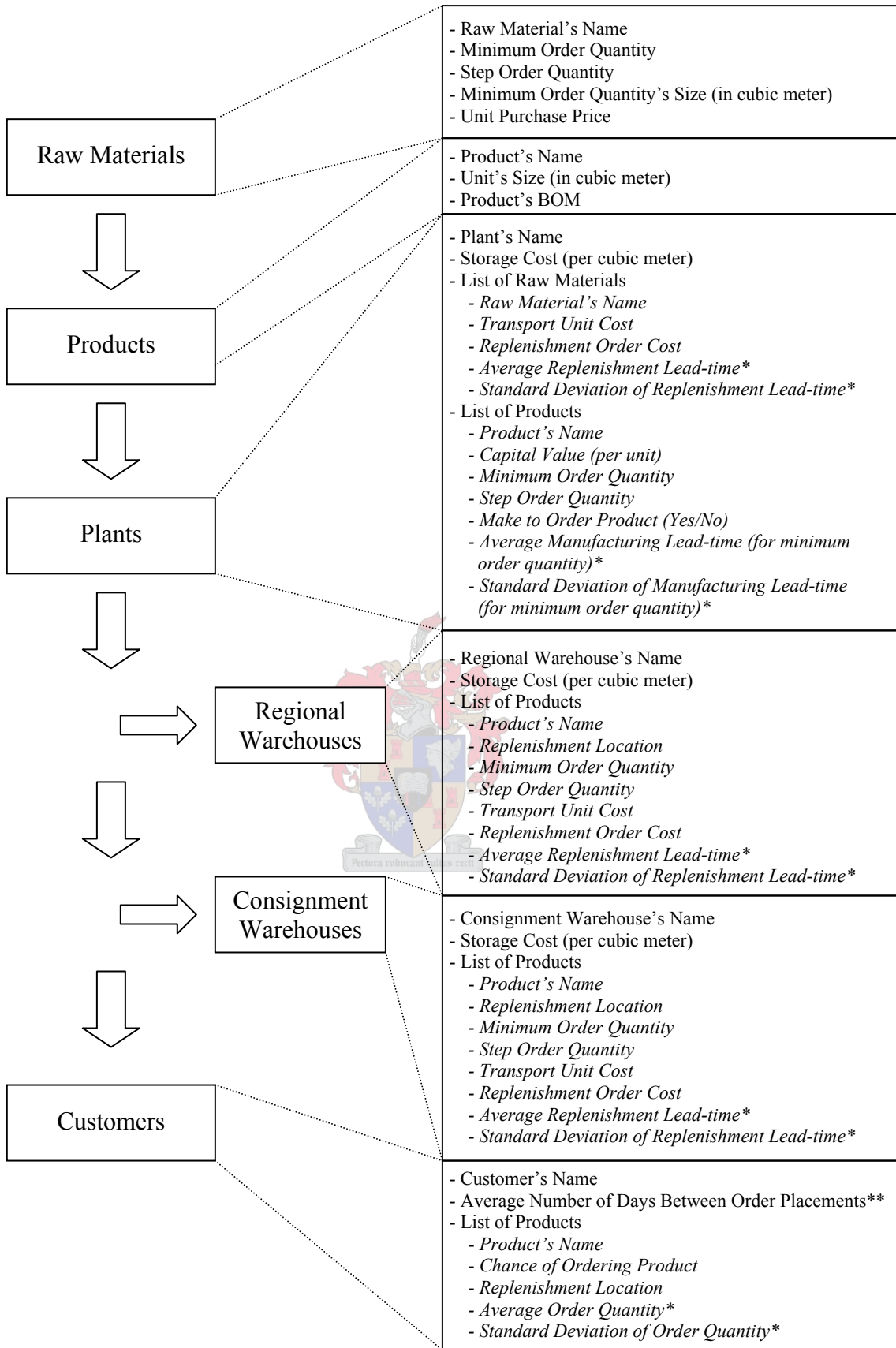
The simulation model consists of six different entities, where these entities are required to model the supply chain configuration as shown in figure 4.1. The entities are: Raw Materials, Products, Manufacturing Facilities or Plants (which consist of a raw material store, plant and a finished goods warehouse), Regional Warehouses, Consignment Warehouses and Customers. The model builder is thus responsible for gathering the data in order to effectively model the above-mentioned entities in the simulation model. The data gathered by the model builder is discussed in the next section.

4.2 DATA REQUIRED BY THE MODEL BUILDER

The model builder is responsible for gathering the data required to model the supply chain configuration, as shown in figure 4.1, in the simulation model. As shown above the simulation model consists of six different entities, namely: Raw Materials, Products, Plants, Regional Warehouses, Consignment Warehouses and Customers. The sequence in which the model builder gathers the data for the various entities, together with the type of data gathered for each entity, is shown in figure 4.2. In order to reduce the amount of complexity modelled, the following assumption was made with regard to the data gathered:

Assumption: Discount on bigger raw material lot sizes is ignored; this implies that the purchase price per unit of a raw material stays constant and does not vary depending on its kanban size.

It should also be noted that for every raw material and product stored in the various locations defined in a supply chain, a minimum order quantity and a step order quantity must be defined. The definition of the minimum order quantity is quite obvious; it purely states the minimum number of units that can be ordered from the upstream supplier of the raw material or product in the supply chain. The step order quantity is the number of units one can incrementally increase the order quantity for the specific raw material or product to. This is best explained in an example:



* Normal Distribution

** Exponential Distribution

Figure 4.2: The sequence in which data is collected by the model builder.

If Product A is ordered on pallets and the minimum number of pallets that must be ordered are two, where each pallet contains 10 units of Product A, then the minimum order quantity for Product A is 20 and the step order quantity is 10. This implies that the order quantity for Product A can be one of the following: 20, 30, 40, etc. The minimum order quantity and step order quantity, together with all the other data gathered by the model builder are stored in the database. The design of the database used to store the data gathered by the model builder, is discussed in the next section.

4.3 DESIGN OF THE DATABASE REQUIRED TO STORE THE MODEL BUILDER'S DATA

The purpose of the database is to store the data required to define an organisation's supply chain in the simulation model and to act as the middleman between the simulation model and the optimisation algorithm, as discussed in Chapter 2. The entity relationship diagram of the database required to store the data gathered by the model builder is shown in figure 4.3, while table 4.1 shows the various fields stored in each of the tables.

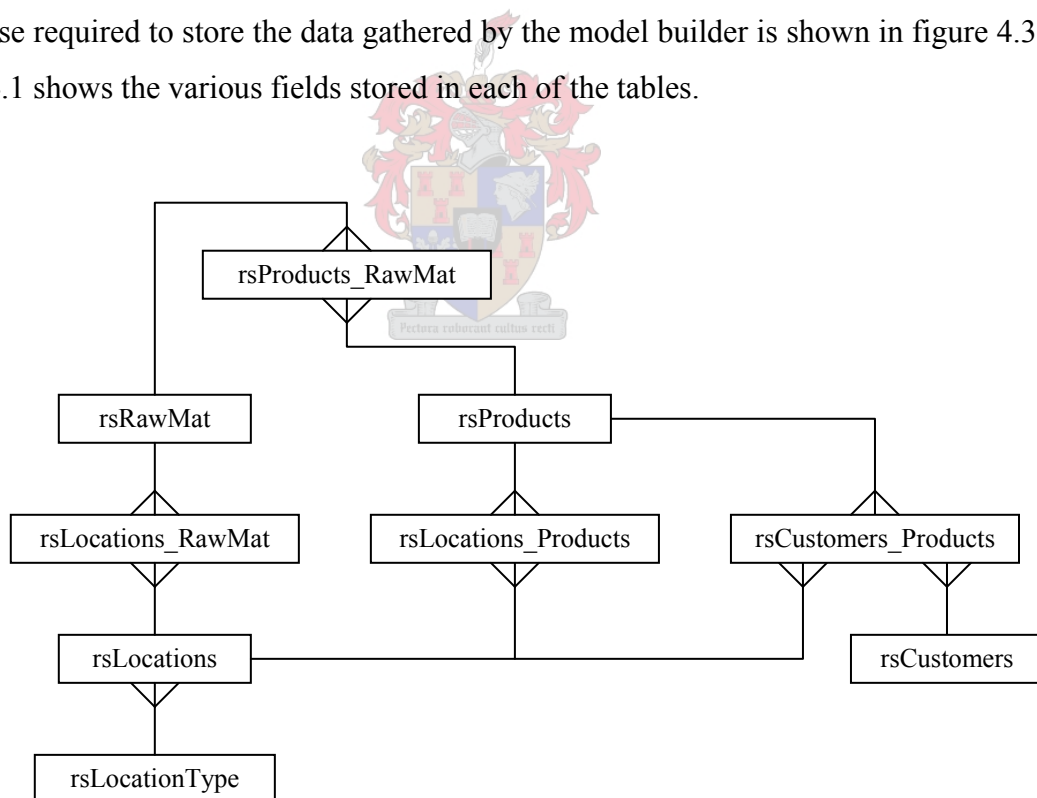


Figure 4.3: The entity relationship diagram of the database for modelling an organisation's supply chain.

Table 4.1: The data fields stored in each of the database's tables.

<i>Table</i>	<i>Fields</i>
rsLocationTypes	<u>LocationType_ID</u> , LocationType
rsLocations	<u>Location_ID</u> , Location_Name, <u>LocationType_ID</u> , StorageCost
rsRawMat	<u>RawMat_ID</u> , RawMat_Name, UnitCost, MinOrderQuantity, StepOrderQuantity, MOQ_Size
rsProducts	<u>Product_ID</u> , Product_Name
rsProducts_RawMat	<u>Product_ID</u> , <u>RawMat_ID</u> , RawMatQuantity
rsLocations_RawMat	<u>Location_ID</u> , <u>RawMat_ID</u> , TransCost, OrderCost, AvgLeadTime, StdDevLeadTime
rsLocations_Products	<u>Location_ID</u> , <u>Product_ID</u> , <u>ReplenshLocation_ID</u> , MakeToOrder, MinOrderQuantity, StepOrderQuantity, CapitalValue, TransCost, OrderCost, AvgLeadTime, StdDevLeadTime
rsCustomers	<u>Customer_ID</u> , Customer_Name, AvgArrival
rsCustomers_Products	<u>Customer_ID</u> , <u>Product_ID</u> , <u>Location_ID</u> , ProductChance, AvgQuantity, StdDevQuantity

The data dictionary containing the various fields stored in each of the database's tables, as shown in table 4.1, is shown in Appendix B. The model builder is however not the only building block in the optimisation tool that utilises the database, and thus influences its design. The simulation model and optimisation algorithm discussed in chapters 5 and 6 respectively also influence its design, where their impact on the database's design is discussed in their respective chapters. In the next chapter the detail design of the simulation model is discussed starting with the concept model of the supply chain modelled within the simulation package, right through to the statistical analysis of the model's output.

CHAPTER 5

THE SIMULATION MODEL

The model builder discussed in Chapter 4 is responsible for gathering all the data required to effectively model an organisation's supply chain. The data gathered by the model builder is used by the simulation model to simulate the stochastic processes occurring within the defined supply chain. Implementing the objective function discussed in Chapter 3 within the simulation model enables the comparison of various inventory solutions against each other, with the purpose of determining the optimum inventory solution for a defined supply chain. Chapter 5 is thus concerned with the development of the simulation model in its totality.

Chapter 5 is structured as follows: The use of simulation in supply chain optimisation is briefly discussed in 5.1. This is followed by the definition of the concept model to be implemented in a simulation software package in 5.2. The parameters to be investigated are discussed in 5.3, while 5.4 is concerned with the implementation of the concept model in a simulation software package. The analysis of the output parameters of the model is discussed in 5.5 and the requirements placed on the database's design in 5.6.

5.1 THE USE OF SIMULATION IN SUPPLY CHAIN OPTIMISATION

Simulation is flexible and powerful enough to model any dynamic system in which one element of the system may dramatically influence the operation and performance of another element in the system. Because simulation models accurately reproduce the logic and random effects of a system, it is used extensively in the improvement and optimisation of supply chains world wide.

The most common use of simulation in supply chain optimisation is the modelling of a specific supply chain, or part of a chain, with the purpose of analysing and improving the chain. This was and still is successfully done for many supply chains, some examples are: a liquid natural gas (LNG) supply chain [Stchedroff & Cheng 2003, pp. 1608-1611], a railroad coal transportation system [Franzese et al. 2003, pp.1602-1606] and a computer assembly factory, where it includes the factory's inbound and outbound logistics [Jain & Choong 2002,

pp. 1165-1173]. This is however a very time consuming process, since the supply chain needs to be analysed and conceptualised before a model can be built in a simulation software package. After the simulation model has been built it still needs to be verified and validated before any analysis can be done using the model. Currently the focus is on reducing the time it takes to develop a simulation model.

In trying to reduce a simulation model's development lead-time various objects required to simulate a supply chain have been defined. These objects can then be used as building blocks when constructing a new simulation model. An example of such building blocks is the work done by Hamoen and Moens [2002, pp. 1315-1218], where they defined building blocks required to model any steel production factory's logistics. Similar work has been done by Rossetti and Chan [2003, pp. 1612-1620], where they defined simulation objects that can be used in developing more general supply chain configurations.

There are also various software packages on offer, such as Supply Chain Builder and TRICEPS Real-Time Warehouse Management [ITtoolbox Supply Chain Knowledge Base, 2004]. These packages allow the user to define his supply chain or a section of the chain with the purpose of analysing it and thus improving it. It should be noted that software packages still only cater for subsets in the whole supply chain spectrum and not the entire spectrum itself i.e. specific type- or industry specific supply chains are catered for by the software.

This thesis can be seen as part of this simulation-optimisation group, where the focus is specifically on the modelling of Just-In-Time (JIT) supply chains with the purpose of determining the optimum inventory levels. The concept model required to effectively model a JIT supply chain is discussed next.

5.2 THE SUPPLY CHAIN CONCEPT MODEL

The simulation concept model states the various decision processes followed in a supply chain regarding the handling of inventory, and can be divided into four subgroups, namely:

- Order handling at the various facilities in the supply chain.
- Manufacturing of product.
- Replenishment of raw materials.
- Receiving of replenishment orders at each facility in the supply chain.

The facilities modelled in the simulation model are:

- Consignment warehouses.
- Regional warehouses.
- Plant warehouses.
- Plants.
- Raw material warehouses.

The boundary of the simulation model is shown in figure 4.1 on page 19, where it includes the raw material replenishment lead-times, but excludes the suppliers of raw materials. The travel lead-time from any location in the supply chain to a customer is also omitted in the model, since the study is concerned with finding the optimum inventory level whilst the travel lead-time cannot be manipulated by means of inventory. In the simulation model the customers act as triggers to the system by placing orders at the various locations defined in the supply chain.

The concept model of the supply chain is shown in figures 5.1 to 5.8, where figures 5.1 to 5.3 are concerned with the handling of orders placed at the consignment, regional and plant warehouses. Figure 5.4 states the method in which production orders are handled at plants in the supply chain. The generation and reception of raw material replenishment orders are shown in figure 5.5, while reception of stock replenishment orders at the plant, regional and consignment warehouses are shown in figures 5.6 to 5.8 respectively.

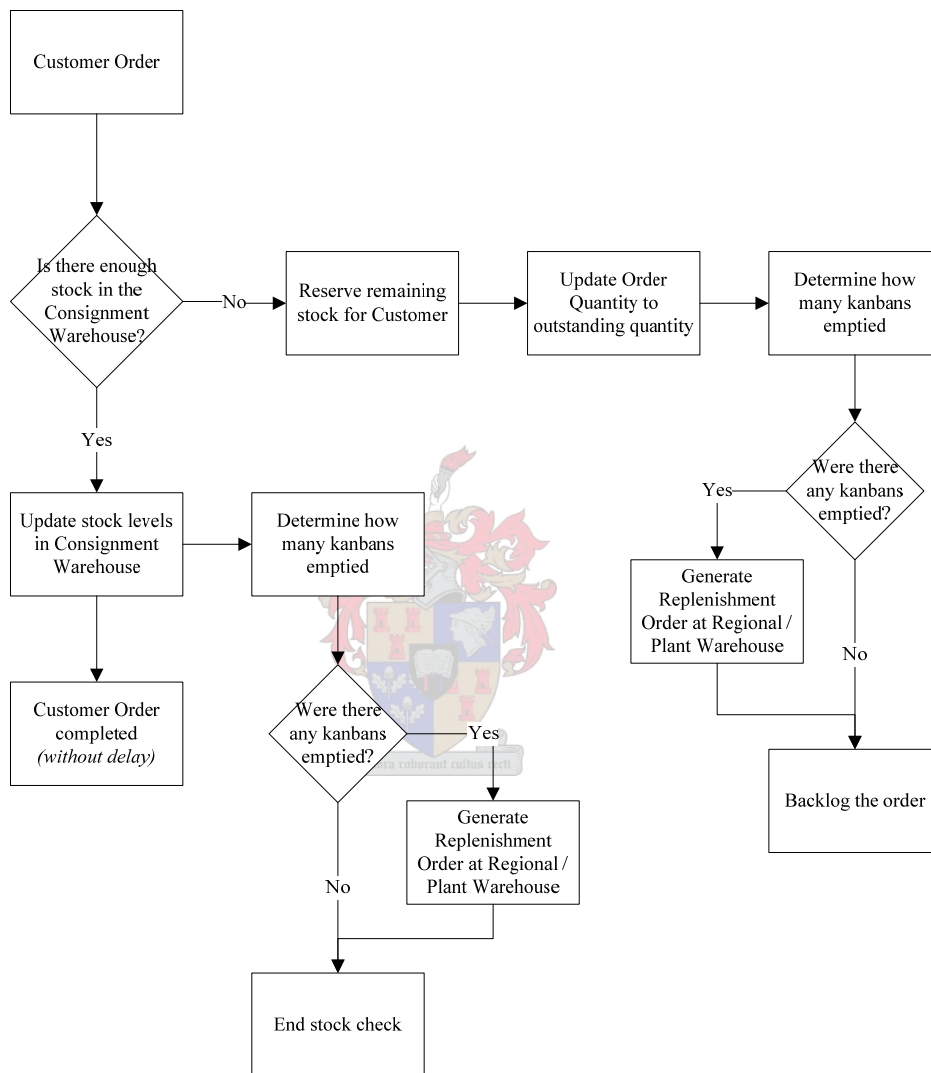


Figure 5.1: The handling of customer orders placed at a consignment warehouse.

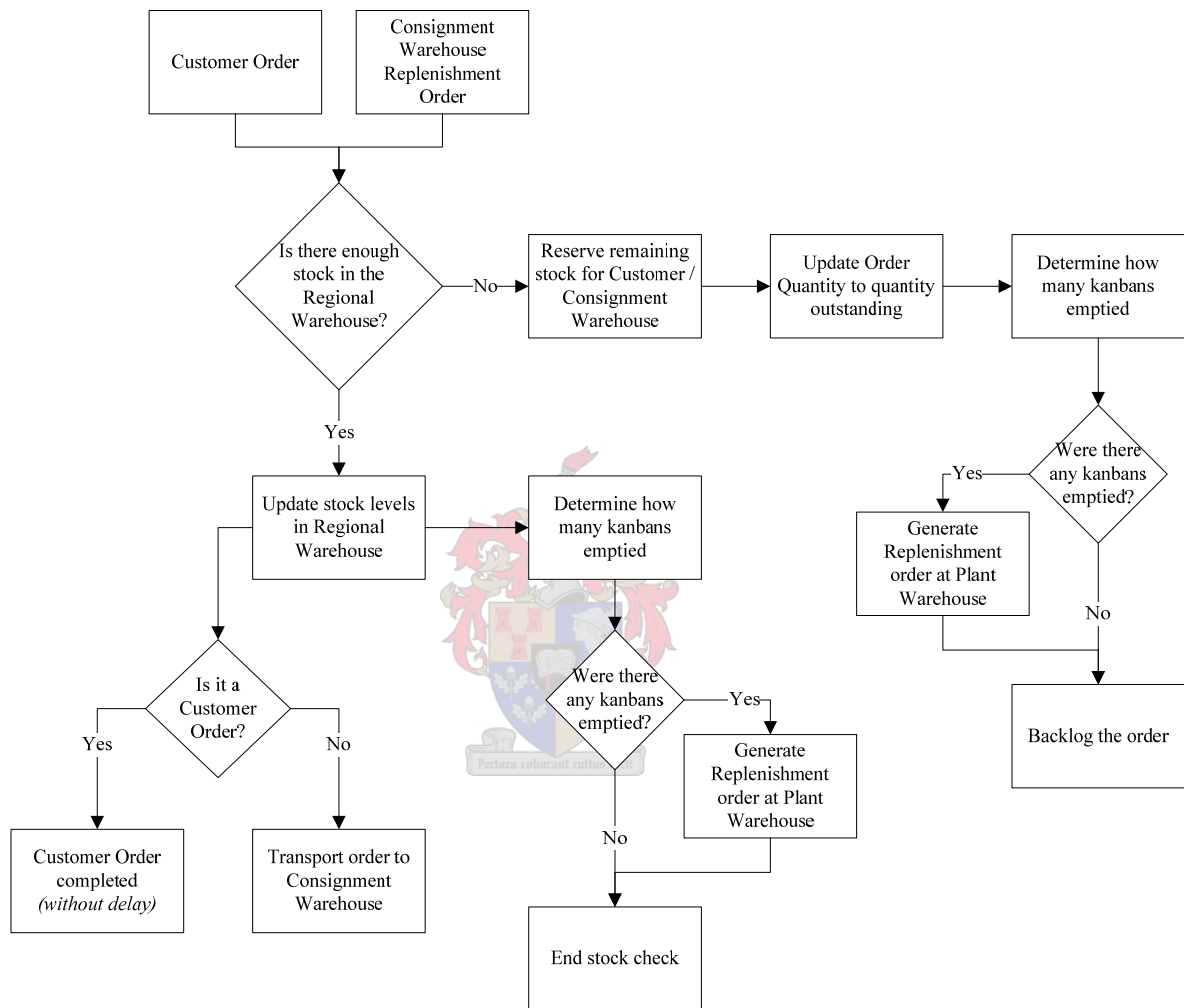


Figure 5.2: The handling of customer orders and stock replenishment orders placed at a regional warehouse.

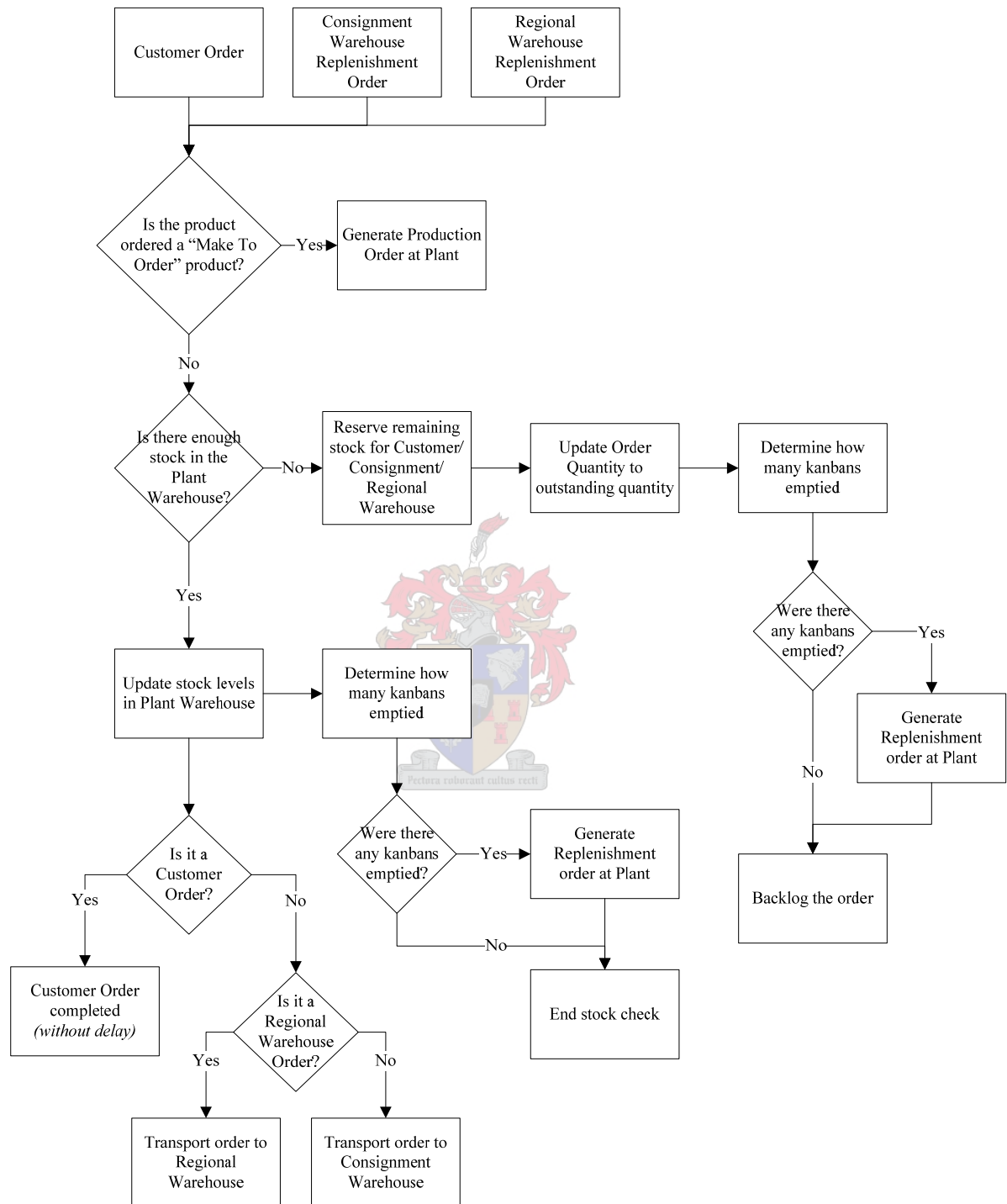


Figure 5.3: The handling of customer orders and stock replenishment orders placed at a plant warehouse.

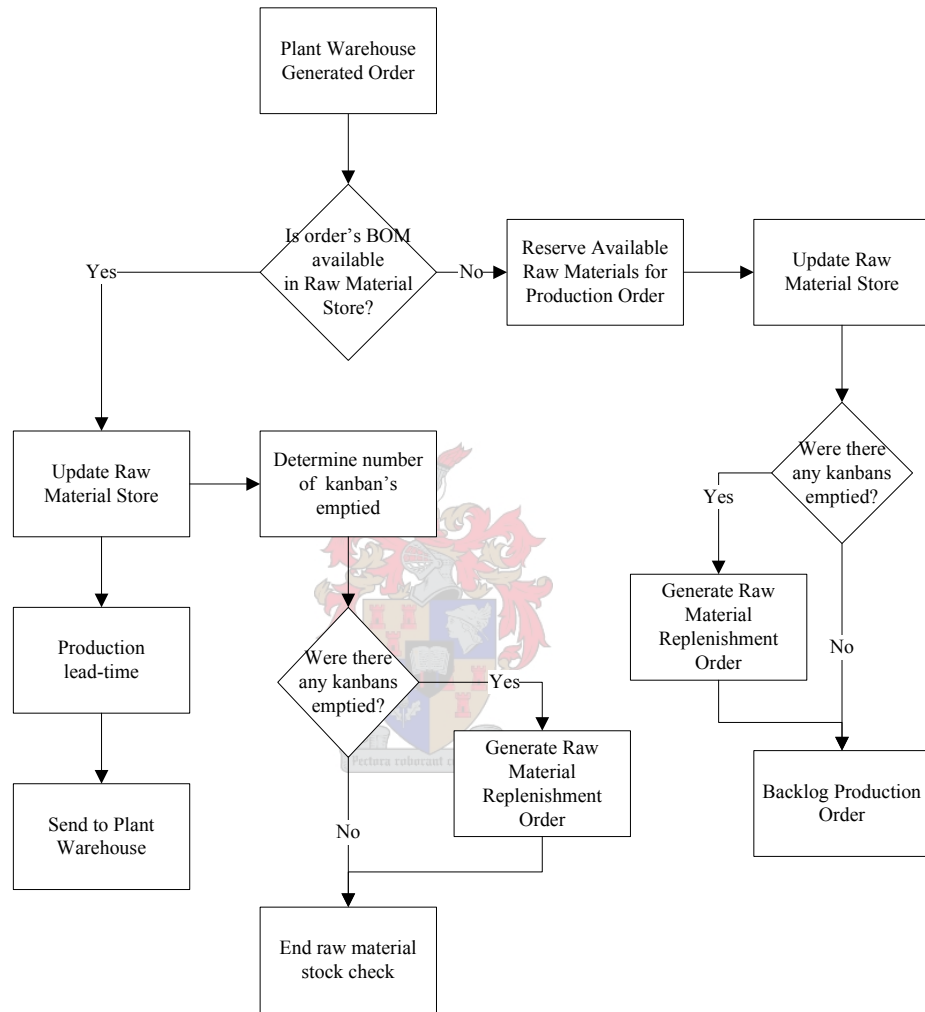


Figure 5.4: The handling of production orders placed at a plant.

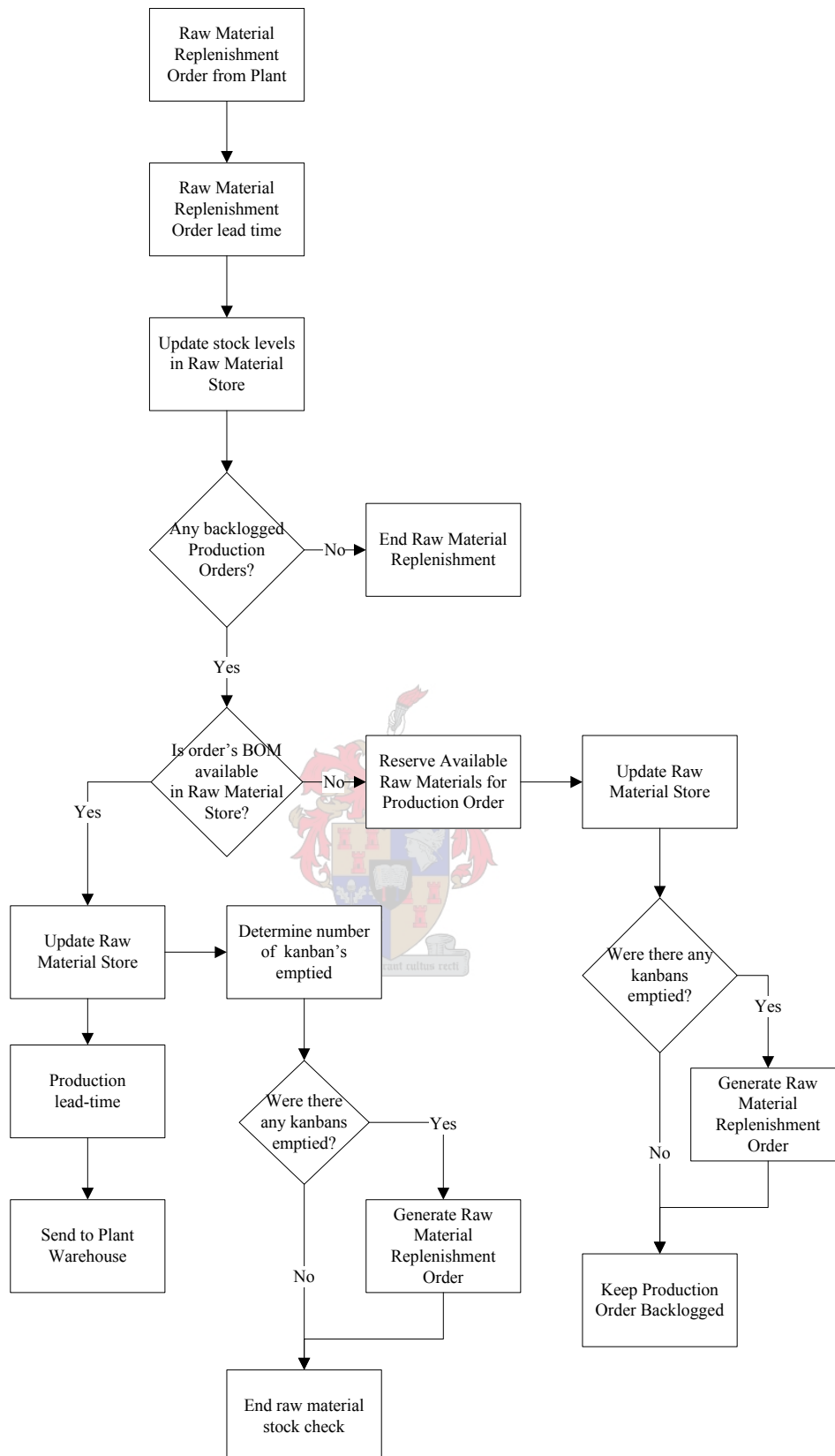


Figure 5.5: The ordering and receiving of raw material replenishment orders at a plant's raw material warehouse.

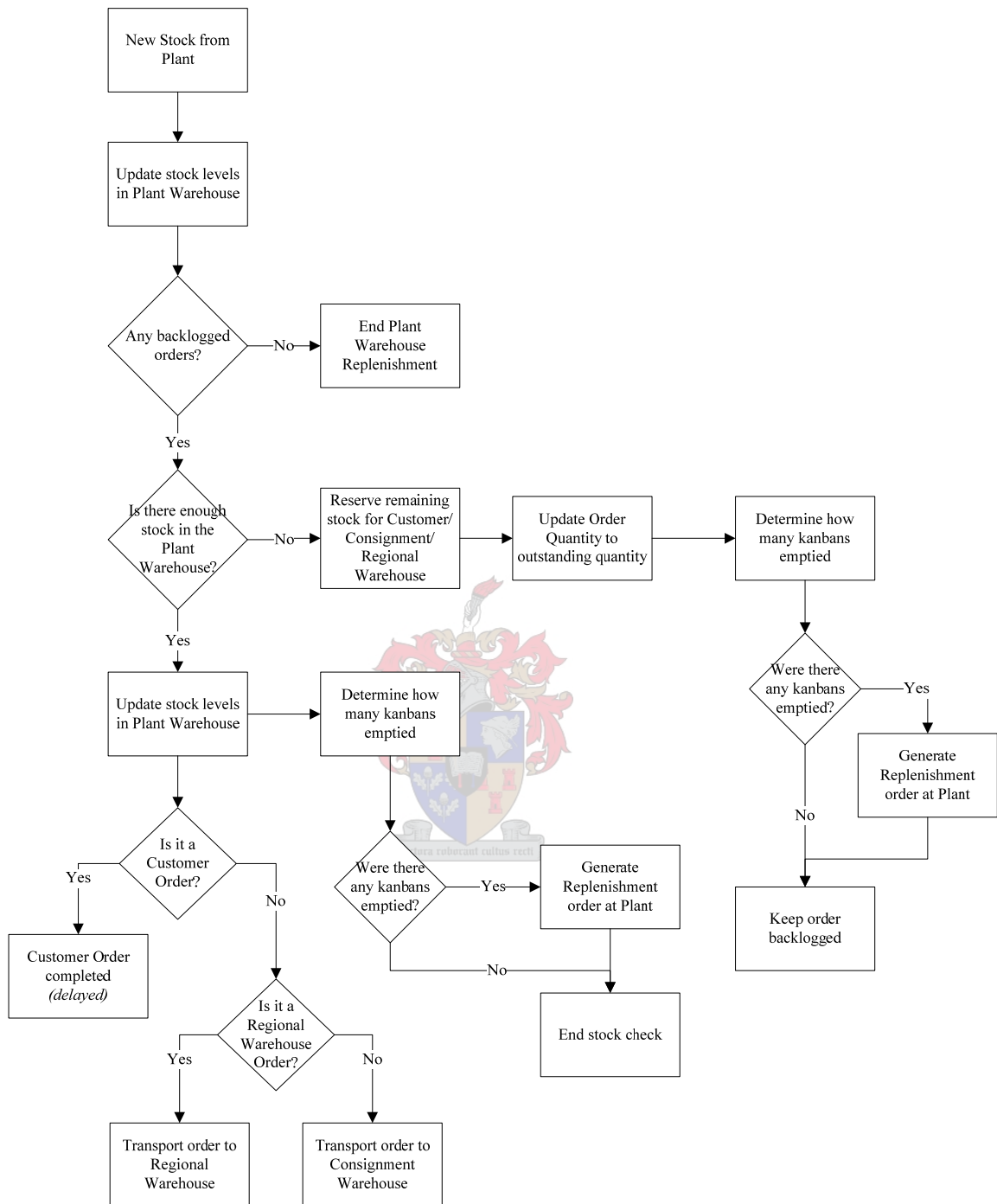


Figure 5.6: The process followed at a plant warehouse when a production order is booked to the warehouse.

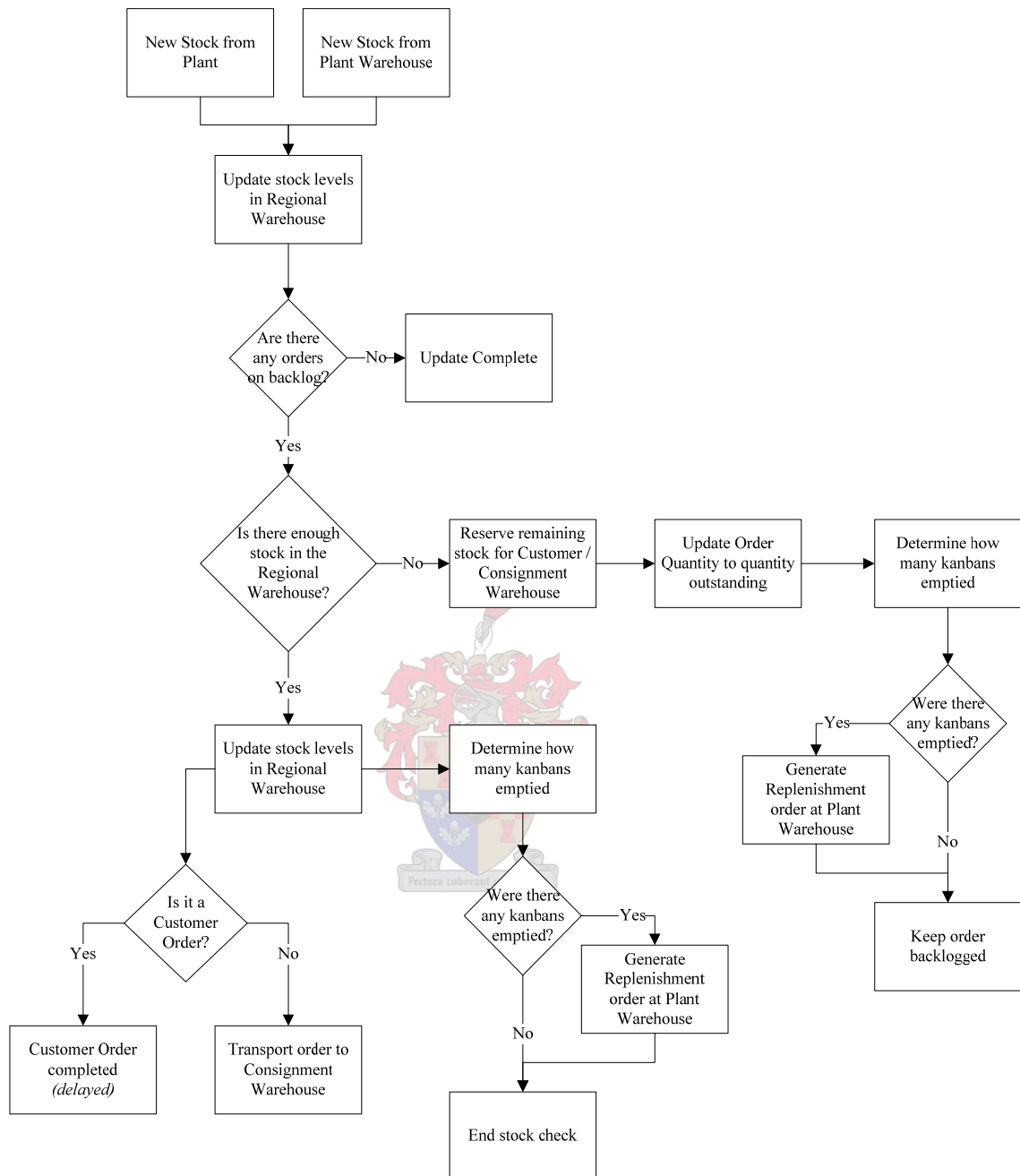


Figure 5.7: The process followed at a regional warehouse when a stock replenishment order is received from an upstream location in the supply chain.

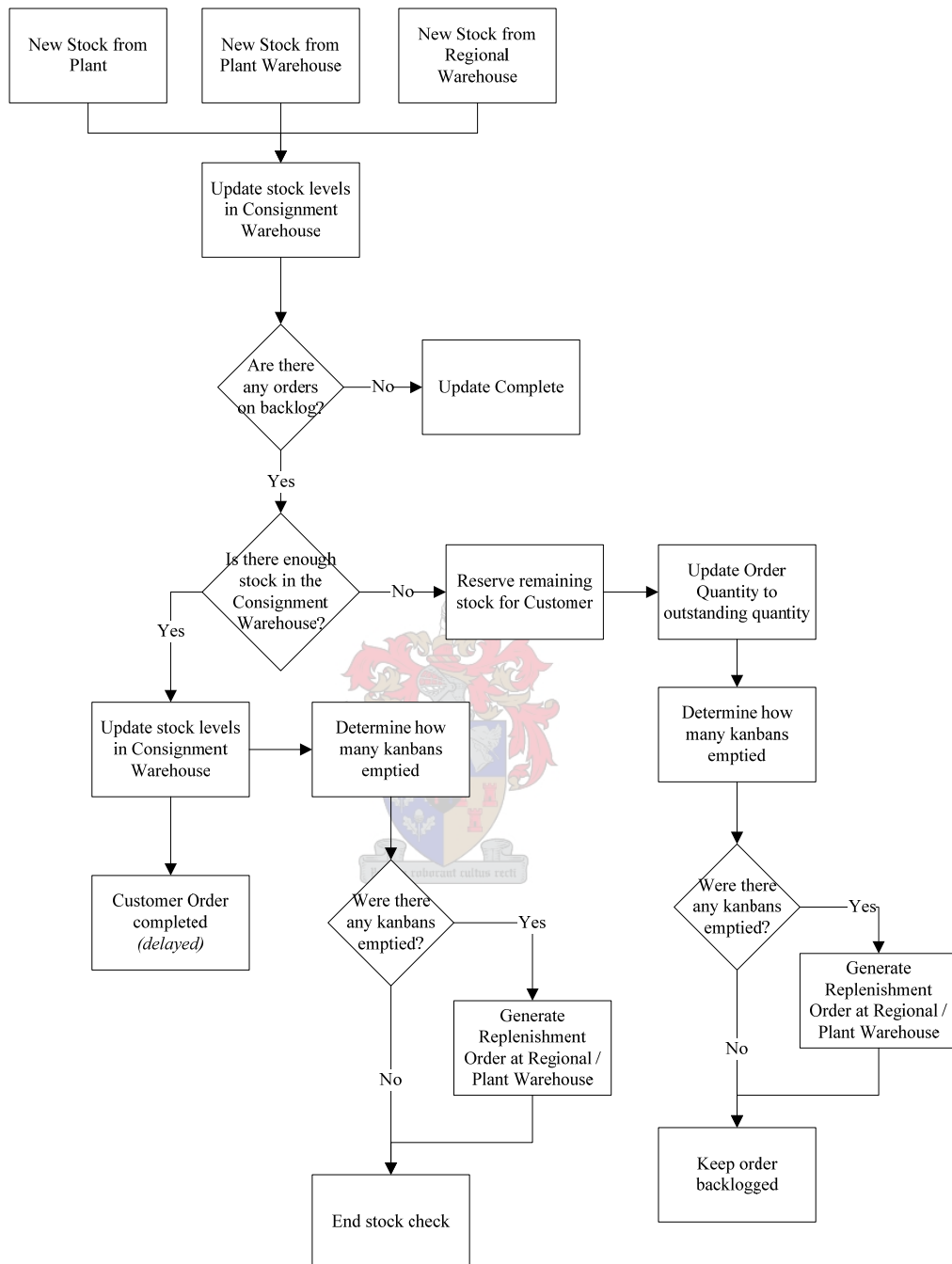


Figure 5.8: The process followed at a consignment warehouse when a stock replenishment order is received from an upstream location in the supply chain.

The following assumptions are made in the concept model with regard to the manufacturing of products:

Assumption 1: There is no capacity constraint in any of the plants defined within the supply chain.

Assumption 2: There is no minimum production order quantity for make-to-order products, since the quantity ordered is governed by the order quantity distributions of the customers (as discussed in Chapter 4).

Assumption 3: In the event of a raw material shortage, (in other words there are not enough raw materials to fully complete a production order) the plant will reserve the available raw materials for the production order and backlog the order until the required raw materials arrive.

The current section defined the concept model to be implemented in the simulation model, while the next section defines the parameters to be investigated by the simulation model with the purpose of comparing various inventory solutions against each other.

5.3 PARAMETERS TO BE INVESTIGATED

The purpose of the simulation model is to simulate the stochastic processes within a defined supply chain. This is done to evaluate the performance of a possible inventory solution, and ultimately compare the performance of various solutions against each other. The objective function discussed in Chapter 3 is used to determine the performance of each inventory solution.

The objective function, as shown in equation 3.10 on page 17, consists of four components:

- Opportunity cost of capital invested in inventory.
- Storage cost.
- Transport cost.
- Replenishment order cost.

It further has a predefined customer service level as a constraint condition to which any valid inventory solution must adhere.

In the implementation of the objective function within the simulation model only the parameters whose values are influenced by the stochastic processes occurring within the supply chain are implemented in the model. Therefore the parameters to be studied in the simulation model are:

- Transport cost.
- Replenishment order cost.
- Customer service level.

The opportunity cost of the capital invested in inventory and the storage cost of inventory are omitted from the simulation model, since their values are known before every simulation run. This is due to the assumptions made in 3.3.2.1 and 3.3.2.2 on pages 13 and 14 respectively. It is important to note that they are not ignored in the optimisation process, they are determined before each simulation run and used in the analysis of every proposed inventory solution after each simulation run. (Refer to figure 6.3 on page 60.)

Having defined the simulation concept model and the parameters to be investigated by the simulation model, the translation of the concept model to a computer model will be discussed in the next section.

5.4 TRANSLATION OF CONCEPT MODEL TO COMPUTER MODEL

As stated in Chapter 1, the final deliverable of the thesis is a generic JIT supply chain optimisation tool, whereby an organisation can determine the optimum stock levels throughout its supply chain. The optimisation tool should therefore be able to cater for various supply chain configurations, within the boundaries as defined by Chapter 4 and shown in figure 4.1 on page 19.

The concept model discussed in 5.2 states the processes surrounding the handling and management of inventory at each of the entities defined within the supply chain's boundary. The translation of the concept model to a generic computer model is discussed in the following section. The simulation software package used to build the computer model is

briefly discussed in 5.4.1, followed by a discussion on the approach used to build the generic computer model within the selected software package in 5.4.2.

5.4.1 Simulation Software Package Used

With so many simulation software packages on offer, the selection of a software package to build the simulation model in is no trivial task. Table 5.1 states the requirements, together with the optional features, of the simulation software package required to model the generic supply chain.

Table 5.1: The requirements of the simulation software package to model the generic supply chain.

<i>Requirements</i>	<i>Optional</i>
<ul style="list-style-type: none"> • Read and write to a database. • Communication with external programs. • Easy implementation of user defined decision logic. 	<ul style="list-style-type: none"> • Animation. • Statistical input and output analysis.

The data required to model an organisation's supply chain is stored in the database, as discussed in Chapter 4. It is thus critical that the generic simulation model can communicate with the database and therefore the simulation package must be able to read and write to a database.

Another requirement of the simulation software package is the interfacing of the package with external programs. This is mainly due to the optimisation process, where the optimisation algorithm prompts the simulation model to run a simulation, and analyses the output of the simulation after each run. This process is continued until the optimum inventory solution is determined by the optimisation algorithm; it is therefore of the utmost importance that the simulation software package can communicate with external software packages, since the optimisation algorithm is implemented as an external program to the simulation model.

The simulation software package should not be rigid in the way it handles the entities within a simulation model. The user should be able to easily define the decision logic to which the entities in the simulation model must be subjected. This will ensure that the concept model discussed in 5.2 on page 26 can be implemented fully, and with ease, in the simulation software package.

The software package need not be strong in animation, since it will be running in the background while the optimisation algorithm acts as the user interface. Similarly the simulation software package need not possess an input-output analyser, since the analysis of each simulation run's output is handled by the optimisation algorithm, while the analysis of the input to the model is done by the user prior to using the tool.

There is however a host of simulation software packages satisfying all the requirements stated in table 5.1. The deciding factor in the selection of a simulation package is the cost of acquiring the package itself. The package selected is Simul8, since the cost of acquiring the package is much less compared to other simulation packages and the funds available for the project are limited. Simul8 fulfils all of the requirements stated in table 5.1:

- Simul8 connects to a database via an ODBC connection, and reads and writes to a database using SQL queries.
- Communication between Simul8 and external programs is achieved using COM objects.
- The *Visual Logic* programming language of Simul8 allows the user total freedom with the implementation of the decision logic required to implement the concept model shown in figures 5.1 to 5.8 on pages 27 to 34. [Albertyn & Kruger 2003, p. 58]

The translation of the concept model to a computer model in Simul8 is discussed in detail in the following section.

5.4.2 Translation to a Generic Computer Model

The simulation model must be generic to ensure that the optimisation tool caters for all possible supply chain configurations, since a supply chain can consist of any number of each of the following location types: consignment warehouses, regional warehouses and plants. In knowing that an organisation's supply chain is totally defined within the database, as

discussed in Chapter 4, the simulation model only needs to draw from the database in order to simulate an organisation's supply chain, i.e. the simulation model is defined by the data in the database for any organisation. It is thus not necessary to model each and every location in a supply chain, but rather the decision processes followed at each type of location as defined in the concept model discussed in 5.2 on page 26. Thus any order received at any one of the three location types defined within the simulation model, whether it is a customer- or replenishment order, determines the specific location that the location type must simulate at that instance in a simulation run.

In order to model the three location types, the simulation model is divided into six sections namely:

- Order Generation
- Consignment Warehouse
- Regional Warehouse
- Plant Warehouse
- Plant
- Raw Material Replenishment.

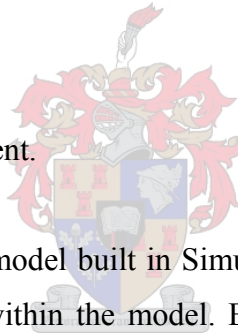


Figure 5.9 shows the simulation model built in Simul8, where it is possible to see the above mentioned sections constructed within the model. Each of these sections is responsible for modelling a location type or part of it, and therefore their designs are discussed in detail in the following sub sections.

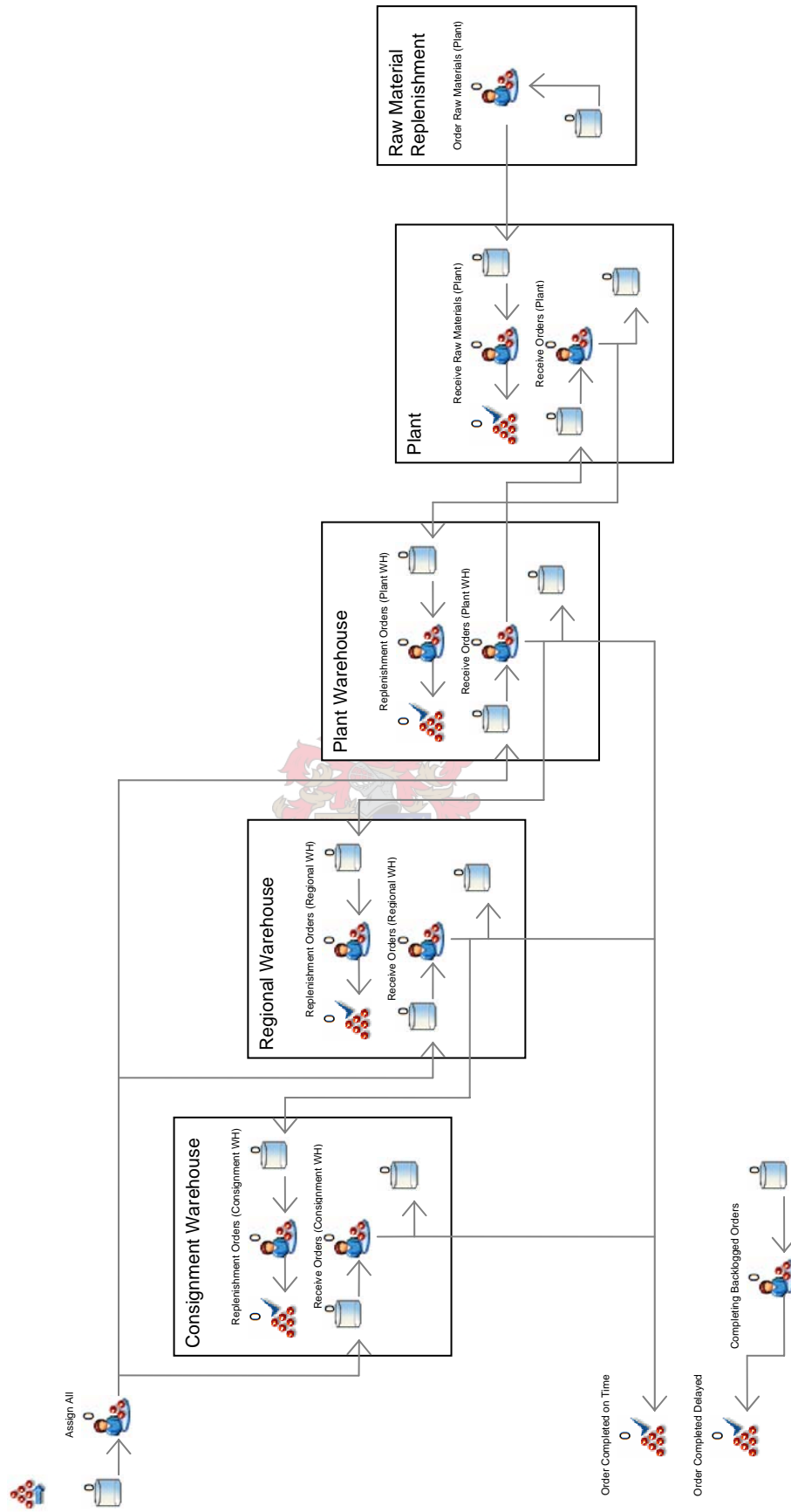


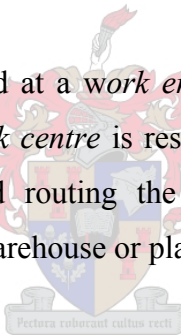
Figure 5.9: The translation of the simulation concept model to a Simul8 computer model.

5.4.2.1 Order Generation Section

The order generation section of the simulation model is responsible for two operations, namely the generation of customer orders and the routing of these orders to their relevant location types. In order to generate customer orders a *work entry point* needs to be added to the simulation model for each customer defined in the database. Each *work entry point* has its own exponential arrival distribution as defined for each customer in the database.

The *work entry points* are however not built in the generic simulation model, but are added to the model by the optimisation algorithm before it starts the optimisation process. Each of the customers' *work entry points* are responsible for assigning the customer's ID (Customer_ID), which is stored in the database, as a label (attribute) to the generated order entity. This is done to assist each of the location types built in the simulation model to model the correct location when an order entity arrives at one of them.

After a customer order is generated at a *work entry point* it is routed directly to the *work centre* termed *Assign All*. This *work centre* is responsible for assigning a product and order quantity to each order entity and routing the entity to its relevant location type i.e. consignment warehouse, regional warehouse or plant warehouse.



The product assigned to the order entity is selected using a random variable. The variable is used in conjunction with the empirical distribution of possible products ordered by the specific customer to assign a product to an order. The customer's product-empirical distribution is compiled from the various products ordered by the customer, where the *ProductChance* field stored in the *rsCustomers_Products* table for each product-customer combination within the database indicates the frequency at which a specific product is ordered by a specific customer. The product's ID (Product_ID) is then read from the database and stored as a label on the order entity. The assigned product ID together with the customer ID determines the parameters for the quantity distribution read from the database, where the distribution is used to assign an order quantity to the order entity as a label. As shown in figure 4.2 on page 21, the quantity assigned to every customer-product combination is normally distributed.

In knowing the customer ID and product ID it is possible to determine the location where the order entity is to be routed to, i.e. the location where the customer places an order for a specific product. The location's ID is read from the database and again stored as a label on the order entity. The location's ID is however linked to a location type's ID stored in the database; it is this location type ID which ultimately determines the routing of the order entity, where it can either be a consignment warehouse, regional warehouse or plant warehouse.

5.4.2.2 Consignment Warehouse Section

The consignment warehouse section of the simulation model is responsible for modelling all the consignment warehouses defined in the supply chain. At any consignment warehouse there are two operations performed, namely the handling of orders placed at the warehouse and the reception of replenishment orders at the warehouse from upstream locations in the supply chain. These operations' flow diagrams are shown in figures 5.1 and 5.8 on page 27 and 34 respectively.

The abovementioned operations are each implemented in the simulation model using a *work centre*, where these *work centres* are termed *Receive Orders (Consignment WH)* and *Replenishment Orders (Consignment WH)* respectively.

Each of the customer order entities arriving at the consignment warehouse section contains a product ID and location ID as labels, as discussed in 5.4.2.1. These two IDs allow the *Receive Orders (Consignment WH) work centre* to model a specific consignment warehouse defined in the database. A customer order received at a consignment warehouse can either then be completed or backlogged depending on the order size and the stock at hand at that location. If there is enough stock the order is routed to the *work exit point* responsible for completing customer orders by the *work centre*. Conversely backlogged orders are routed to a *storage area*, where all the backlogged order entities are stored until enough stock is available to complete them.

As a result of customer orders placed at consignment warehouses, replenishment orders are generated at upstream locations in the supply chain, when a kanban of stock is consumed. The location types of the upstream locations from the consignment warehouse are the regional and

plant warehouses. A replenishment order is generated by creating a new order entity at the required replenishment location type. The customer ID label of the newly-created order represents the current consignment warehouse ID. (A flag label termed *replenishment* is set to indicate that the specific order entity is a replenishment order and not a customer order.) The *Receive Order (Consignment WH) work centre*, which is responsible for handling replenishment orders in the consignment warehouse section, uses the customer ID label together with the product ID label to determine which specific consignment warehouse to model on receiving a replenishment order.

The *Replenishment Orders (Consignment WH) work centre* is also responsible for removing backlogged orders from the *storage area* keeping the backlogged order entities. When a replenishment order of a specific product is received the backlogged queue is scanned for orders of that specific product at that specific location. This is done to determine if one or more orders can be removed from the queue and thus be completed. If this is the case the order entity is removed from the *storage area* and routed to the *work exit point* responsible for completing backlogged orders, otherwise the entity remains in the *storage area*.

5.4.2.3 Regional Warehouse Section

The regional warehouse section of the simulation model is very similar to that of the consignment warehouse section; it is modelled using two *work centres*, where one *work centre* is responsible for handling orders placed at a regional warehouse and the other for receiving replenishment orders at a regional warehouse. These *work centres* are termed *Receive Orders (Regional WH)* and *Replenishment Orders (Regional WH)* respectively, where the flow diagram of each of the above mentioned processes is shown in figures 5.2 and 5.7 on pages 28 and 33 respectively.

The only differences between the regional warehouse and the consignment warehouse sections in the simulation model are:

- Customer orders together with replenishment orders are placed at a regional warehouse.
- The regional warehouse's replenishment orders are only generated at plant warehouses.

Thus orders placed at a regional warehouse can be completed if it is a customer order and there is available stock, routed back to a consignment warehouse if it is a replenishment order and there is stock available or backlogged if there is not enough stock. The *Receive Orders (Regional WH) work centre* thus has the added responsibility of handling replenishment orders, when comparing it to that of the consignment warehouse section's *work centre*.

If the *Receive Orders (Regional WH) work centre* is to send a replenishment order back to its originator, the defined delivery lead-time distribution for the specific product sent to its replenishment location, needs to be assigned to the order entity. This is achieved using the order entity's customer ID and product ID labels to locate the specific product-location combination in the database. The delivery lead-time distribution is then read from the database and loaded into the simulation model, from where the lead-time is assigned to the specific order entity.

Similarly backlogged orders consist of customer orders and replenishment orders stored in the backlog *storage area* of the regional warehouse section. If a replenishment order is received and there are backlogged orders, the orders are removed from the queue similar to those in the consignment warehouse section. If there is however a replenishment order, the order is routed back to the downstream location who ordered it, by assigning it a delivery lead-time instead of completing the order as in the case with a customer order.

5.4.2.4 Plant Warehouse Section

The plant warehouse section of the simulation model is again a derivation from the regional warehouse section. It consists of two *work centres* termed *Receive Orders (Plant WH)* and *Replenishment Orders (Plant WH)*, where these *work centres* are responsible for the handling of orders placed at a plant warehouse and the receiving of replenishment orders from the plant. The flow diagrams stating the decision process at each of the abovementioned *work centres* are shown in figures 5.3 and 5.6 on page 29 and 33 respectively.

In the plant warehouse section the replenishment orders can only be placed at the plant section defined in the simulation model, while orders can be received from customers directly, the consignment warehouse section and the regional warehouse section. There is no difference in

the implementation of the plant warehouse section to that of the regional warehouse section, apart from the handling of orders for make-to-order products placed at a plant warehouse.

The plant warehouse section thus has the added responsibility of routing make-to-order product orders directly to the plant section, and also to route the finished production orders received from the plant section back to the order's originator by assigning the correct delivery lead-time to each order.

5.4.2.5 Plant Section

The purpose of the plant section is to model any plant warehouse's plant and raw material store, where figures 5.4 and 5.5 on pages 30 and 31 shows the flow diagram of the various processes involved. The plant section is implemented in the simulation model similar to all the abovementioned sections, using two *work centres*. These *work centres* are *Receive Orders (Plant)* which is responsible for the handling of all the production orders placed at a plant (figure 5.4 on page 30), and *Receive Raw Materials (Plant)* which is responsible for booking new raw materials into the raw materials store (figure 5.5 on page 31).

The *work centre* responsible for receiving production orders placed at a plant, uses the product ID label of the order entity to find the specific product's bill of materials (BOM) in the database. If all the raw materials are available to manufacture the production order, the parameters of the manufacturing lead-time distribution are determined by converting the minimum production order quantity's lead-time distribution, stored in the database, to a lead-time distribution resembling the order quantity. The manufacturing lead-time for all products are normally distributed as shown in figure 4.2 on page 21. The conversion is shown in equation 5.1, where the function *ROUND* rounds any number to the nearest integer:

$$\mu_{ORDER} = ROUND \left[\frac{Q_{ORDER}}{Q_{MIN}} \right] \mu_{MIN}$$

$$S_{ORDER} = S_{MIN} \sqrt{ROUND \left[\frac{Q_{ORDER}}{Q_{MIN}} \right]}$$
...(5.1)

Where:

μ_{ORDER} – The average number of days to complete the current production order.

S_{ORDER} – The standard deviation of the current production order.

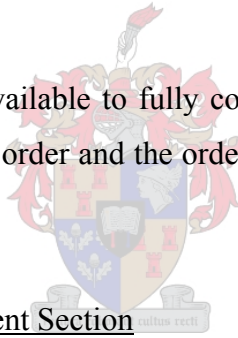
Q_{ORDER} – The order quantity of the current production order.

Q_{MIN} – The minimum production order quantity stored in the database.

μ_{MIN} – The average number of days to complete the minimum production order quantity (as stored in the database).

S_{MIN} – The standard deviation of for the minimum production order (as stored in the database).

If not all the raw materials are available to fully complete a production order, the available raw materials are reserved for the order and the order is backlogged until the outstanding raw materials are replenished.



5.4.2.6 Raw Material Replenishment Section

The purpose of the raw material replenishment section is to simulate the replenishment lead-time of each raw material ordered from the various plant warehouses in a supply chain. This is done using the location ID and product ID stored on the order entity as labels. In this case the order entity is a replenishment order, where the product ID represents the raw material's ID and the location ID the order location's ID, both of which are stored in the database. The raw material-location combination is then used to retrieve the parameters of the lead-time distribution from the database, from where it is loaded in the simulation model. The replenishment lead-time for all raw materials are normally distributed as shown in figure 4.2 on page 21.

As mentioned previously, the purpose of the simulation model is to model the stochastic processes occurring in a supply chain, with the goal of comparing various inventory solutions

against each other. This is achieved by implementing the objective function discussed in Chapter 3 with the parameters to be investigated discussed in 5.3 on page 35. The analysis of these parameters is discussed in the following section.

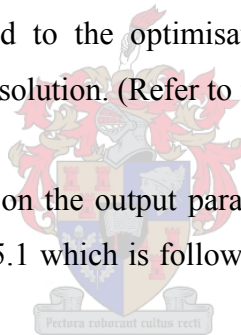
5.5 ANALYSIS OF SIMULATION OUTPUT

The output parameters of the simulation model are:

- Transport cost.
- Replenishment order cost.
- Customer service level.

After each simulation run, the abovementioned parameters are analysed, with one of the following two results: The simulation's run length is extended to ensure an acceptable confidence level half-width is achieved for each of the parameters investigated, or the output parameters' results are forwarded to the optimisation algorithm where they are used to determine the optimum inventory solution. (Refer to figure 2.1 on page 3.)

Before any analysis can be done on the output parameters an initial simulation run must be completed, this is discussed in 5.5.1 which is followed by a discussion on the analysis of the output parameters in 5.5.2.



5.5.1 Initial Simulation Run

The simulation model as defined in 5.2 on page 26 is a non-terminating system, where the simulation time is measured in days. As a warm-up period for a particular run the simulation model is run for 365 days before any of the parameters is recorded. After the warm-up period has been completed, the parameters' results are batched in batches of 365 days until 10 batches have been compiled.

This method of analysis is a simplification of the batch means approach, where the batch means approach seeks to obtain statistical independent observations to make the application of terminating system analysis possible for non-terminating systems. [Bekker 2003, p. 52] In the batch means approach the observations in the warm-up period is discarded and the remaining observations batched in equally sized batches as shown in figure 5.10.

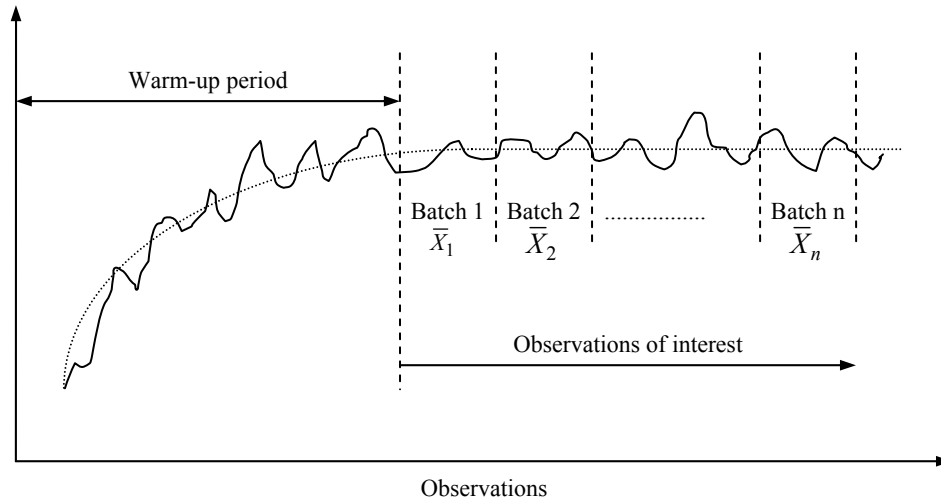
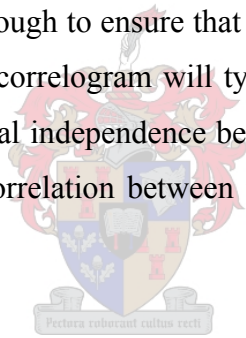


Figure 5.10: Discarding the transient phase (warm-up period) and batching the remaining observations. [Bekker 2003, p. 53]

The selection of the size of the batches in the batch means approach is critical, since the batches' size needs to be large enough to ensure that the batches are statistically independent. In a once-off simulation study a correlogram will typically be used to determine the size of the batches thus ensuring statistical independence between batches, where a correlogram is a graphical representation of the correlation between the various observations made. [Bekker 2003, p. 57]



In the optimisation tool the optimisation process is automated, thus it is not possible to evaluate a correlogram for each simulation run. The following assumption is therefore made with regard to the size of the batches used in evaluation of a simulation run's output parameters:

Assumption: The lead-times in the organisation's supply chain are short enough to ensure that batches of 365 days are not correlated and are thus statistically independent.

These batches are then analysed to determine whether the simulation run should be extended, or the simulation run terminated and the result passed-on to the optimisation algorithm.

5.5.2 Statistical Analysis of Output Parameters

The deciding factor on whether a simulation's run length must be extended or not, is the confidence level half-width achieved for the various output parameters, where a 95% confidence level (CI) is used for all calculations done in the optimisation tool. The confidence interval half-width for each of the output parameters is determined as follows [Devore & Farnum 1999, p. 297]:

$$h = \frac{t_{n-1, 1-\frac{\alpha}{2}} S}{\sqrt{n}} \quad \dots(5.2)$$

Where:

S – Sample standard deviation.

n – The number of batches.

t – The upper critical $1-\alpha/2$ percentile on the t-distribution with $n-1$ degrees of freedom.

α – Level of significance, typically 5%.

The average of all the batches is determined as follows:

$$\bar{X} = \frac{\sum_{i=1}^n X_i}{n} \quad \dots(5.3)$$

The sample standard deviation in equation 5.2 is determined from the sample variance:

$$S^2 = \frac{\sum_{i=1}^n (X_i - \bar{X})^2}{n - 1} \quad \dots(5.4)$$

Where:

n – The number of batches.

X_i – Average value of batch i .

\bar{X} – The average of all the batches.

As stated in 5.5.1, there are initially 10 batches ($n = 10$) of 365 days each, for each of the output parameters measured in the simulation model. For 10 batches and $\alpha=5\%$, equation 5.2 gives:

$$h = \frac{2.262 \times S}{\sqrt{10}} \quad \dots(5.5)$$

After these 10 batches have been accumulated, the confidence level half-width for each of the output parameters is determined using equation 5.5. Knowing the achieved confidence interval half-width for each of the output parameters, each output parameter is subjected to the following equation. This is done to determine whether the simulation run length should be extended and, if this is the case, to what length it should be extended to. [Bekker 2003, p. 47]

$$n^* = n \left(\frac{h}{h^*} \right)^2 \quad \dots(5.6)$$

Where:

- n – The number of batches used in the mini simulation (10 in this case).
- h – The determined confidence interval half-width of the output parameter.
- h^* – The desired confidence interval half-width.
- n^* – The required number of batches.

The output parameter with the highest n^* determines the required simulation run length to ensure the desired confidence interval half-widths are achieved for the various output parameters. The desired confidence interval half-width for each of the output parameters is shown in table 5.2.

Table 5.2: Desired confidence half-width values (h^*) for the parameters under study.

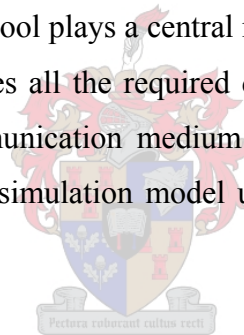
<i>Output Parameter</i>	<i>Confidence Interval Half-width</i>
Customer Service Level	2.5%
Transport Cost	10% of Average Transport Cost
Replenishment Cost	10% of Average Replenishment Cost

If the highest n^* calculated is smaller than n the simulation is stopped, which results in the output parameters being passed on to the optimisation algorithm, otherwise the simulation run length is extended to n^* batches. New estimators for the parameters are determined when the simulation run has terminated.

The database is responsible for defining the supply chain within the simulation model, as has been shown throughout Chapter 5. The database however plays another key role in the optimisation tool; it acts as the communication medium between the optimisation algorithm and the simulation model. The database's design, as discussed in Chapter 4, must thus be extended to accommodate the simulation output parameters. The following section covers all the additional requirements the simulation model places on the design of the database.

5.6 SIMULATION MODEL'S DATABASE REQUIREMENTS

The database of the optimisation tool plays a central role in facilitating the whole optimisation process, where the database stores all the required data which fully defines a supply chain. The database also acts as communication medium between the simulation model and the optimisation algorithm, plus the simulation model uses the database as an extension of the simulation model.



To enable the database to act as the communication medium between the simulation model and the optimisation algorithm, an extra table called *rsResults* is added to the database, where figure 5.11 shows the entity relationship diagram of the revised database design.

The new table is responsible for storing the customer service level, transport cost and replenishment order cost for each batch compiled during a simulation run. The customer service level is determined from the table using the following two fields: *GoodNr* and *TotalNr*, where *GoodNr* stores the sum of the order quantities immediately dispatched to customers and *TotalNr* the sum of all the order quantities ordered by customers for a particular batch in a simulation run. (Refer to equation 3.4 on page 8.) Conversely the transport cost and replenishment order cost are tallied-up for each batch and stored in *TransCost* and *ReplenishCost* respectively.

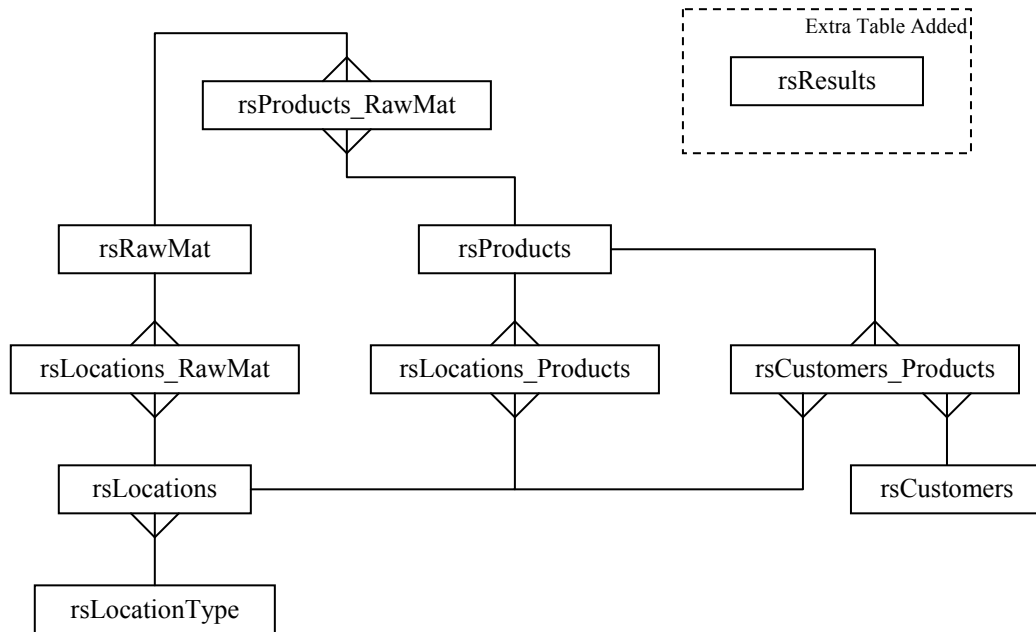


Figure 5.11: The revised entity relationship diagram.

The database's design is further extended by adding four more fields to the following two tables: *rsLocations_RawMat* and *rsLocations_Products*. The four fields added are: *KanbanSize*, *KanbanNumber*, *Stock* and *Orig_KanbanNumber*. These fields, with the exception of *Orig_KanbanNumber*, are used by the simulation model to keep track of each product's and raw material's stock level at every location defined in the supply chain.

The *Stock* field is used to keep track of the number of units in stock of each product at each location; if product is removed or replenished at a location in the simulation model, the quantity added or removed is either subtracted or added to the *Stock* field.

On the other hand the *KanbanSize* and *KanbanNumber* fields are purely used for the triggering of stock replenishment orders from upstream locations in the supply chain, where a replenishment order is generated if the following statement is true:

$$\frac{Stock}{KanbanSize} \leq KanbanNumber - 1$$

The *Orig_KanbanNumber* field is not used by the simulation model, but stores the original kanban number assigned by the optimisation algorithm, for a particular simulation run, to

each product and raw material at each location in the supply chain. Table 5.3 shows all the tables, together with their respective fields, of the extended database. The complete data dictionary of the database is shown in Appendix B.

Table 5.3: The database's tables together with their respective fields.

<i>Table</i>	<i>Fields</i>
rsLocationTypes	<u>LocationType_ID</u> , LocationType
rsLocations	<u>Location_ID</u> , Location_Name, <u>LocationType_ID</u> , StorageCost
rsRawMat	<u>RawMat_ID</u> , RawMat_Name, UnitCost, MinOrderQuantity, StepOrderQuantity, MOQ_Size
rsProducts	<u>Product_ID</u> , Product_Name
rsProducts_RawMat	<u>Product_ID</u> , <u>RawMat_ID</u> , RawMatQuantity
rsLocations_RawMat	<u>Location_ID</u> , <u>RawMat_ID</u> , TransCost, OrderCost, AvgLeadTime, StdDevLeadTime, <i>KanbanSize</i> , <i>KanbanNumber</i> , <i>Stock</i> , <i>Orig_KanbanNumber</i>
rsLocations_Products	<u>Location_ID</u> , <u>Product_ID</u> , <u>ReplenshLocation_ID</u> , MakeToOrder, MinOrderQuantity, StepOrderQuantity, CapitalValue, TransCost, OrderCost, AvgLeadTime, StdDevLeadTime, <i>KanbanSize</i> , <i>KanbanNumber</i> , <i>Stock</i> , <i>Orig_KanbanNumber</i>
rsCustomers	<u>Customer_ID</u> , Customer_Name, AvgArrival
rsCustomers_Products	<u>Customer_ID</u> , <u>Product_ID</u> , <u>Location_ID</u> , ProductChance, AvgQuantity, StdDevQuantity
<i>rsResults</i>	<i><u>Batch_ID</u>, GoodNr, TotalNr, TransCost, ReplenishCost</i>

* The fields in italics are the fields added to the various tables in the database when compared to the database designed in Chapter 4. (Table 4.1 on page 23)

The whole purpose of the simulation model is to simulate the stochastic processes that occur within a supply chain. This is done to test the various inventory solutions proposed by the optimisation algorithm, where the objective function as defined in Chapter 3 (Refer to equation 3.10 on page 17) is used to distinguish the various solutions from each other. The optimisation algorithm selected and the implementation thereof are discussed in the following chapter.

CHAPTER 6

THE OPTIMISATION ALGORITHM

The supply chain optimisation tool is compiled of various building blocks, as shown in figure 2.1 on page 3. Each of the building blocks is responsible for specific tasks in the optimisation tool, as has been shown for the database, model builder and the simulation model in chapters 4 and 5. The optimisation algorithm and the report generator are the remaining two building blocks which need to be discussed.

In the optimisation tool, the optimisation algorithm is responsible for governing the optimisation process, i.e. finding the optimum inventory solution for the supply chain defined in the database. In the optimisation tool, the optimisation algorithm interacts directly with the database, simulation model and the report generator. (Refer to figure 2.1 on page 3.)

Conversely the report generator is triggered by the termination of the optimisation tool and is only responsible for exporting the solution declared optimum by the algorithm to Microsoft Excel. (Refer to Appendix A, figure A.7 on page A-6, for an example of the report generator's output.) The rest of the chapter is thus concerned with the implementation of the optimisation algorithm within the supply chain optimisation tool, starting with the selection of the optimisation algorithm.

6.1 THE SELECTION OF THE OPTIMISATION ALGORITHM

The selection of the optimisation algorithm is equally important to the development of the objective function and simulation model, where all three of these components have a major influence on the final inventory solution proposed by the optimisation tool. The optimisation algorithm is responsible for finding the optimum inventory solution using the objective function defined in Chapter 3.

The solution space of a defined supply chain is all the possible combinations of kanban numbers and kanban sizes, for all the items in stock at each location in the supply chain. Each of these combinations is a possible (optimum) solution for a supply chain, where the objective

function defined in Chapter 3 quantifies the solution and enables the optimisation algorithm to compare various solutions against each other. (Refer to equation 3.10 on page 17.)

The result of the objective function for all the possible combinations i.e. the solution space, is discrete. This is due to the kanban number defined for each item, where the result of the new solution cannot be predetermined when the number of kanbans defined for a product changed from X to $X-1$. There is also a high probability that the solution space for a defined supply chain contains local optima. Thus the objective function falls into the *NP*-hard (non deterministic polynomial time-hard) problems group, where an optimal solution can only be determined if all the possible combinations have been tested. [Bekker 2004a, p. 407] Metaheuristics are generally used to find near-optimum solutions for the *NP*-hard problems. Thus the optimisation algorithm employed to optimise the inventory levels will fall in the metaheuristic algorithm group.

The metaheuristic algorithm group is filled with a host of possible algorithms, such as simulated annealing, population-based incremental learning (PBIL) and genetic algorithms (GA). The algorithm implemented in the optimisation tool, must however be able to easily adapt to any supply chain configuration, i.e. the optimisation algorithm must be generic for all possible supply chain configurations within the boundaries defined in Chapter 4.



The optimisation algorithm selected from the heuristic group is the Harmony Search Algorithm (HSA), since the working of the algorithm is not complex and it uses integer values instead of binary as is the case with PBIL and GA algorithms. [Geem et al. 2001, p. 61] This implies that the HSA can be implemented in the optimisation tool with ease, ensuring all possible supply chain configurations together with different solution space sizes will be catered for. The working of the HSA is discussed next.

6.2 THE HARMONY SEARCH ALGORITHM

The Harmony Search Algorithm (HSA) is not derived from natural phenomena as is the case with many other heuristic algorithms, such as the genetic and simulated annealing algorithms, but rather from an artificial phenomenon: The harmonisation of instruments. [Geem et al. 2001, p. 62] The HSA mimics an ensemble of instruments, where the musicians experiment with a combination of pitches until a satisfying harmony is achieved. The comparison

between optimisation and musical performance is shown in table 6.1, while the working of the HSA is shown using pseudocode in figure 6.1.

Table 6.1: Comparison between optimisation and musical performance. [Geem et al. 2001, p. 62]

<i>Comparison Factor</i>	<i>Optimisation Process</i>	<i>Performance Process</i>
Best state	Global Optimum	Fantastic Harmony
Estimated by	Objective Function	Aesthetic Standard
Estimated with	Value of Variables	Pitch of Instruments
Process unit	Each Iteration	Each Practice

```

Generate a random initial Harmony Memory (HM)
Set Counter = 0

While Counter < Termination Criteria
  For each instrument in HM generate a random HMCR
    If HMCR is satisfied
      Select a note from current HM
    Else
      Select a note from instrument range
    End if
  End For

  For each selected note generate a random PAR
    If PAR is satisfied
      Adjust chosen note to neighbouring value
    Else
      Retain note as selected
    End if
  End For

  Evaluate new harmony fitness
  If new harmony is fitter than least fit harmony in HM
    Place new harmony in HM at ranked position
    Set Counter = 0
  Else
    Discard new harmony
    Increment Counter
  End if
End While

```

Figure 6.1: The basic pseudocode for the Harmony Search Algorithm [Bekker 2004b, p.19].

6.2.1 Parameters of the Harmony Search Algorithm

As with most metaheuristic algorithms, the HSA has certain parameters that have to be set before the first iteration of the algorithm is initialised. These parameters are:

1. Harmony Memory (HM) size
2. Harmony Memory Considering Rate (HMCR)
3. Pitch Adjustment Rate (PAR)
4. Termination Criteria

6.2.1.1 Harmony Memory Size

The Harmony Memory (HM) is responsible for storing a number of solutions together with each of their results i.e. the note played by each instrument in the ensemble together with the quality of the harmony achieved. The number of solutions stored in the HM is limited by the Harmony Memory size parameter, where the solutions stored in the HM are ranked from best to worst. If a new harmony is better than the lowest ranked harmony in the HM, the new harmony is added, while the lowest ranked harmony is removed from the HM.

6.2.1.2 Harmony Memory Considering Rate (HMCR)

The HMCR is a value between 0 and 1. This value represents the chance of randomly selecting a note from the HM for a given variable (instrument) when creating a new harmony. A uniformly distributed random number, between 0 and 1, is generated for each note and compared to the HMCR. If the random number is bigger than the HMCR, a note is randomly selected out of the entire solution space for the variable in question. Conversely if the random number is less or equal to the HMCR a note is randomly selected from the HM. Refer to figure 6.1.

6.1.2.3 Pitch Adjustment Rate (PAR)

The PAR parameter acts similarly to the mutation coefficient in genetic algorithms, where its sole purpose is to escape local optima. The parameter takes on a value between 0 and 1, and works independently from the HMCR.

The working of the PAR is best explained with an example: If an instrument can take on any note in the integer range 0 to 10 and it is assigned with a note equal to 5 using the HMCR. A PAR of 0.1 implies that the instrument in question has a 10% probability that its note will be adjusted to its neighbouring values, where these values are 4 or 6. An equal probability is given to the upper or lower neighbouring value in the value set, thus both has a 5% probability of being assigned to the instrument. If the random value is higher than 0.1 the instrument's note does not change and thus stays 5.

6.1.2.4 Termination Criteria

The Termination Criteria resembles the maximum number of iterations allowed for the algorithm to find new harmonies that rank number one in the HM. The convergence of the HSA is discussed in detail in 6.2.2.

6.2.2 Convergence of the Harmony Search Algorithm

The termination of the HSA should ultimately happen when the optimum solution has been determined. Unfortunately the convergence of the HSA towards the optimum solution is not a smooth curve as in the case of the Genetic Algorithm (GA) and Population-Based Incremental Learning (PBIL) algorithm. The HSA rather follows a step-like convergence pattern, as shown in figure 6.2.

In the HSA, the harmony ranked first in the HM is the best solution at any given time during the optimisation process. The selection of the Termination Criteria, as discussed in 6.2.1, is thus very critical in allowing the HSA in finding the optimum solution. If the Termination Criteria is set equal to a few iterations the HSA might terminate before a reasonable near-optimum solution is reached. Conversely the optimum solution might have been reached, but due to selecting a bigger Termination Criteria value the HSA runs unnecessarily long before terminating. The quality of solutions in the HM, together with the values for the HMCR and PAR, also has a role to play in the convergence speed of the HSA.

Knowing the working of the HSA, the implementation of the algorithm within the supply chain optimisation tool is discussed in the following section.

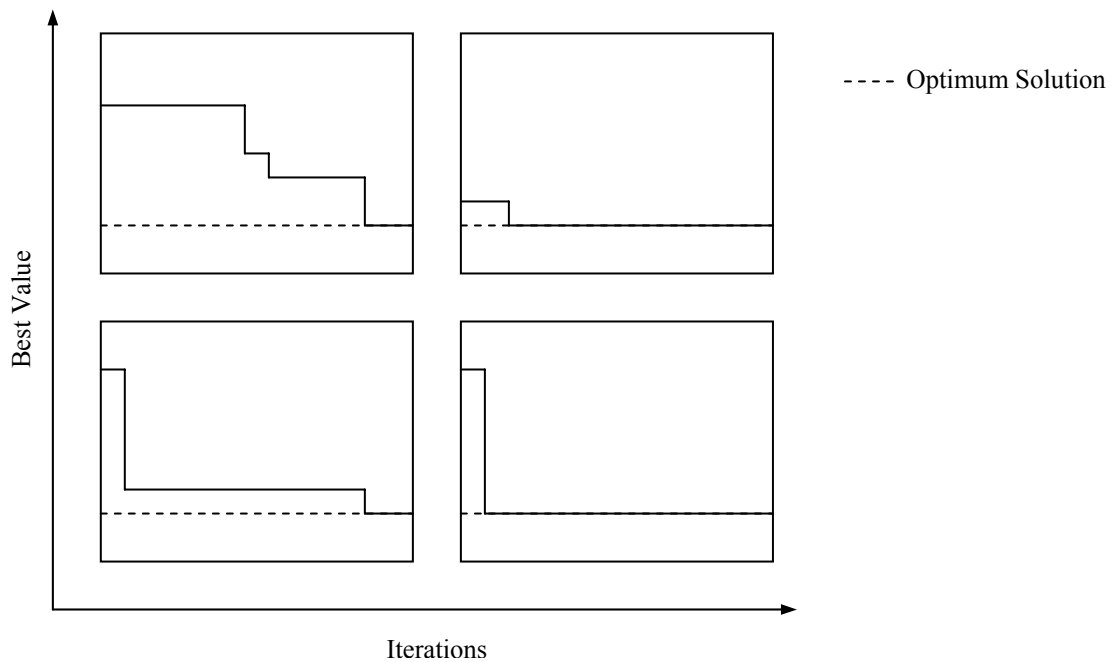


Figure 6.2: Typical convergences patterns of the HSA. [Bekker 2004b, p. 23]

6.3 THE IMPLEMENTATION OF THE HARMONY SEARCH ALGORITHM

During the optimisation process the Harmony Search Algorithm (HSA) interacts with the following three entities in the supply chain optimisation tool: database, simulation model and report generator. In the optimisation tool the HSA is used to minimise the objective function, as it is defined by equation 3.10 on page 17. The working of the optimisation process is shown in figure 6.3, where it is possible to see the interaction between the HSA and the above mentioned entities.

The implementation of the HSA also impacts the design of the database, since the HSA's Harmony Memory (HM) needs to be stored during the optimisation process. Three tables are added to the database to accommodate the HM, these tables are: *rsHarMem*, *rsHarMem_Products* and *rsHarMem_RawMat*. The entity relationship diagram of the redesigned database is shown figure 6.4, while the various fields stored by each table in the database are shown in table 6.1. The data dictionary of the final database is shown in Appendix B.

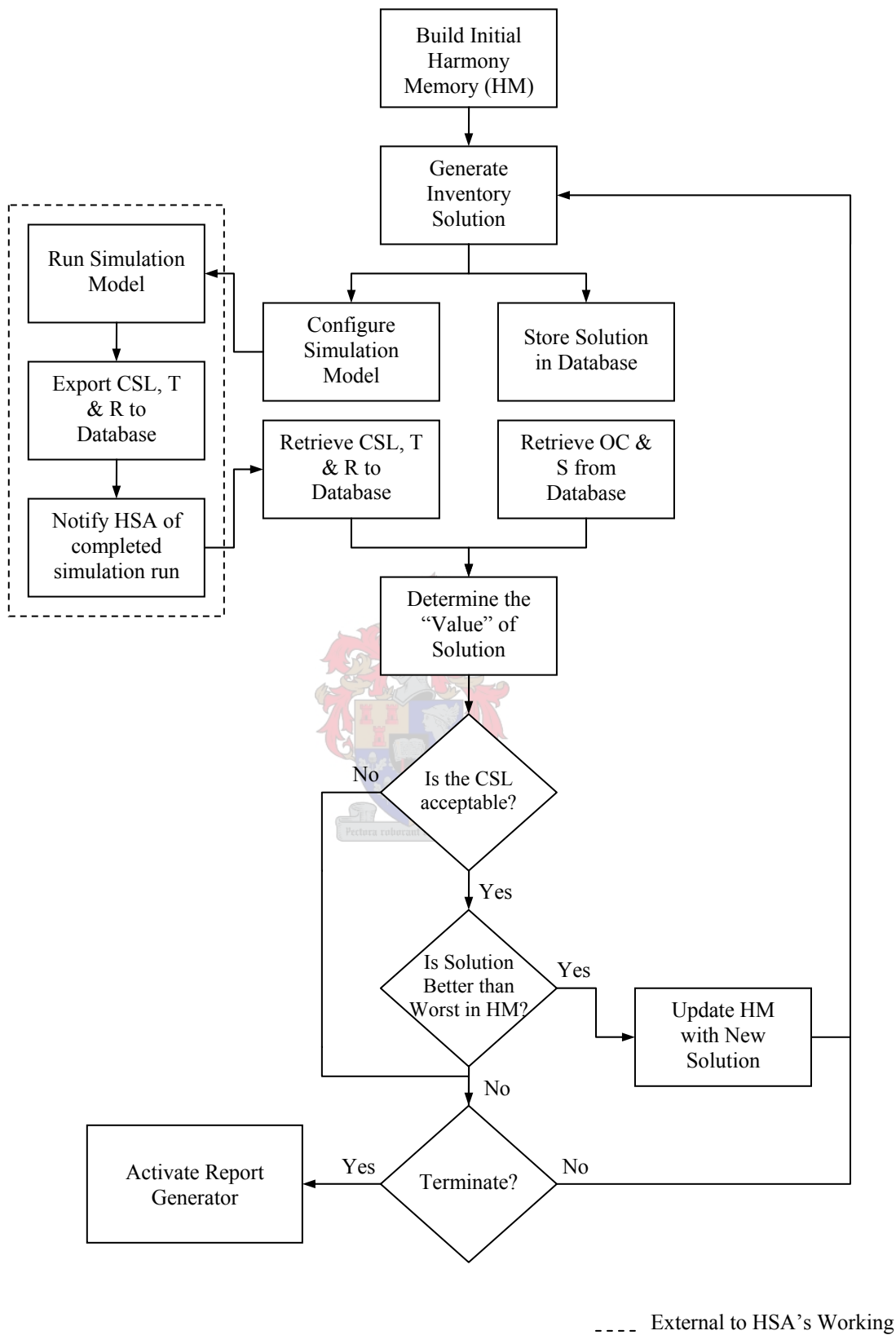


Figure 6.3: The working of the HSA in the optimisation tool.

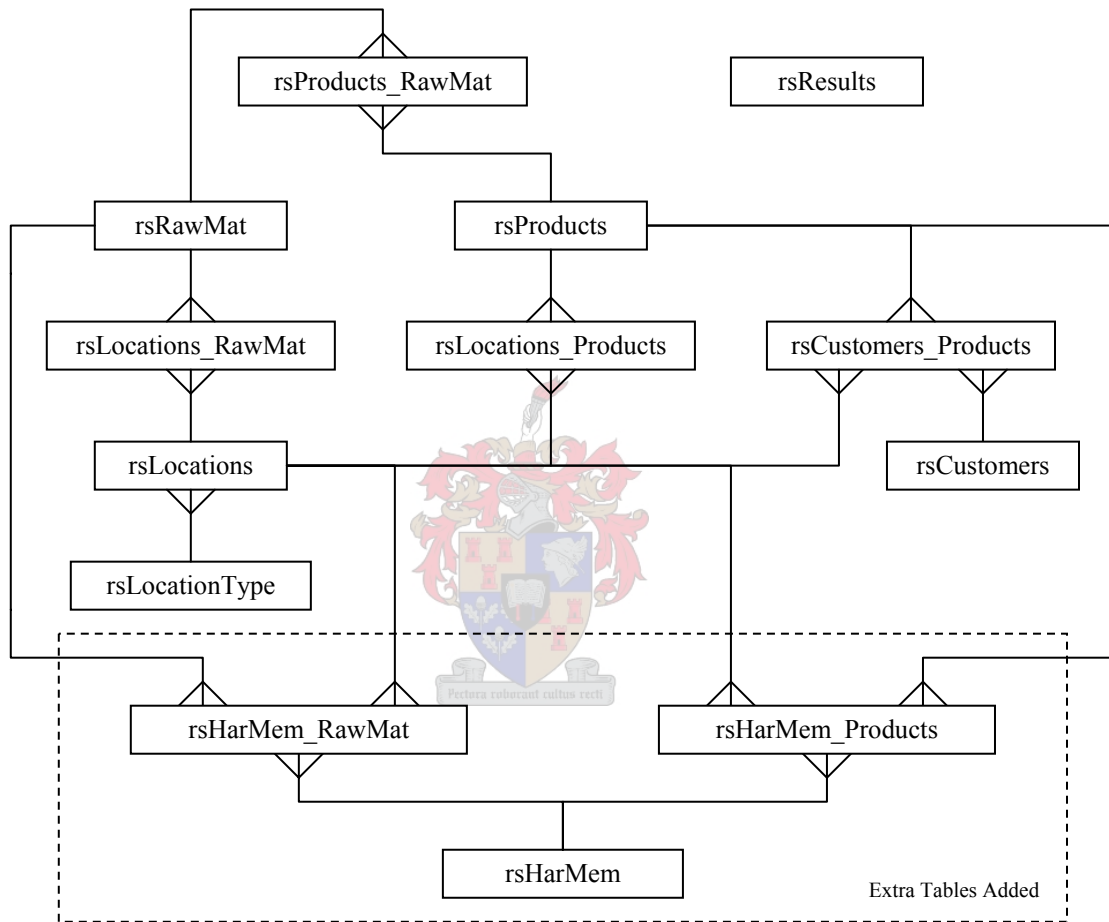


Figure 6.4: The final entity relationship diagram accommodating the HSA’s Harmony Memory (HM).

Table 6.2: The tables and their respective fields, to accommodate the HSA's Harmony Memory (HM).

<i>Table</i>	<i>Fields</i>
rsLocationTypes	<u>LocationType_ID</u> , LocationType
rsLocations	<u>Location_ID</u> , Location_Name, <u>LocationType_ID</u> , StorageCost
rsRawMat	<u>RawMat_ID</u> , RawMat_Name, UnitCost, MinOrderQuantity, StepOrderQuantity, MOQ_Size
rsProducts	<u>Product_ID</u> , Product_Name
rsProducts_RawMat	<u>Product_ID</u> , <u>RawMat_ID</u> , RawMatQuantity
rsLocations_RawMat	<u>Location_ID</u> , <u>RawMat_ID</u> , TransCost, OrderCost, AvgLeadTime, StdDevLeadTime, KanbanSize, KanbanNumber, Stock, Orig_KanbanNumber
rsLocations_Products	<u>Location_ID</u> , <u>Product_ID</u> , <u>ReplenshLocation_ID</u> , MakeToOrder, MinOrderQuantity, StepOrderQuantity, CapitalValue, TransCost, OrderCost, AvgLeadTime, StdDevLeadTime, KanbanSize, KanbanNumber, Stock, Orig_KanbanNumber
rsCustomers	<u>Customer_ID</u> , Customer_Name, AvgArrival
rsCustomers_Products	<u>Customer_ID</u> , <u>Product_ID</u> , <u>Location_ID</u> , ProductChance, AvgQuantity, StdDevQuantity
rsResults	<u>Batch_ID</u> , GoodNr, TotalNr, TransCost, ReplenishCost
<i>rsHarMem</i>	<i><u>HarMem_ID</u>, Score</i>
<i>rsHarMem_RawMat</i>	<i><u>HarMem_ID</u>, <u>RawMat_ID</u>, <u>Location_ID</u>, KanbanNumber, KanbanSize</i>
<i>rsHarMem_Products</i>	<i><u>HarMem_ID</u>, <u>Product_ID</u>, <u>Location_ID</u>, KanbanNumber, KanbanSize</i>

* The fields in italics are the fields added to the various tables in the database when compared to the database in Chapter 5. (Table 5.3 on page 53.)

This chapter concludes the description of the various building blocks from which the supply chain optimisation tool is compiled. In the next chapter the verification and validation of the simulation model and the supply chain optimisation tool as a whole are discussed, together with the various improvements implemented in the tool in order to increase the optimisation speed i.e. the time it takes to reach the optimum solution.

CHAPTER 7

VERIFICATION & VALIDATION

In Chapter 6 the discussion on the design and development of the various components from which the supply chain optimisation tool is compiled was concluded. Chapter 7 is concerned with the methods used in verifying and validating the various components of the optimisation tool and the tool itself. The chapter is divided into two main sections, where the methods used in the verification process of the optimisation tool is discussed in 7.1 while those used in the validation process are discussed in 7.2.

7.1 VERIFICATION OF THE OPTIMISATION TOOL

As shown in Chapter 2 (Refer to figure 2.1 on page 3.) the optimisation tool is compiled from five building blocks, namely: the model builder, database, simulation model, optimisation algorithm and the report generator. The verification of the tool is done by individually verifying the working of each of the above-mentioned building blocks, with special focus on how each component interfaces with the other components. The rest of the section is thus broken up into the following subsections: The verification of the model builder in 7.1.1, the simulation model in 7.1.2, the optimisation algorithm in 7.1.3 and the report generator in 7.1.4. The verification of the database is not done separately, but rather done continuously throughout the verification process since all of the aforementioned building blocks interact directly with the database.

7.1.1 Model Builder

The verification of the model builder is fairly trivial and merely implies the verification of the following two concepts:

1. Ensuring that the data gathered by the model builder is stored correctly in the database i.e. stored in the correct tables and fields defined within the database.

2. Verifying that data validation is done by the model builder, i.e. ensuring that the model builder verifies that the data entered is in the correct format and that all the required fields are completed.

The verification of the first concept implies the inspection of the data stored in the database after it has been entered by the model builder, thus verifying that the data is stored in the correct format, fields and tables within the database. The verification of the second concept is not as straight forward and can never be confirmed fully. The second concept can only be verified above reasonable doubt after the model builder has been tested for a substantial number of hours. During these hours the aim of the user is to try and find program errors and correcting them.

7.1.2 Simulation Model

The verification of the simulation model implies that the software model built in Simul8 adheres to the simulation concept model as discussed in 5.2. (Refer to figures 5.1 to 5.8 from page 27 to 34.) The simulation model's verification process covers the following three areas:

1. Ensuring that the logic of the simulation concept model is implemented in the simulation model.
2. Verifying that the various calculations performed in the model are correct.
3. Verifying that the data read from the database are the correct data.

The verification process is divided into two sections: Verification using a deterministic supply chain model with fixed values for the inter-arrival times, transport and replenishment lead-times and the assignment of order quantities. This is followed by the same supply chain model, but instead of being deterministic a stochastic model is used with its respective normal and exponential distributions as defined by figure 4.2 on page 21. Both the models used to verify the simulation model are shown in Appendix C.

7.1.2.1 Verification using a Deterministic Supply Chain Model

The verification of the simulation model's logic (Refer to figures 5.1 to 5.8 on pages 27 to 34.) is done using a deterministic supply chain model, where the inter-arrival times, transport

and replenishment lead-times and the assignment of order quantities are all drawn from fixed values. It is thus possible to predict the behaviour of the model and to verify the actual behaviour of the model according to its prediction. The check sheet used to predict and track the model's behaviour is shown in Appendix D.

Refer to the check sheet shown in Appendix D: In the first seven days of the simulation run customers arrived according to their arrival distributions as shown in Table C.2 on page C-2. Each customer placed orders for their respective products at the defined locations as shown in Figure C.1 and Table C.4 on pages C-1 and C-2 respectively. The quantity ordered by each customer correlated to what is stated in Table C.3 on page C-2.

On day eight of the simulation run Customer #1 ordered three units of Product B at Regional Warehouse #1. This reduced the total number of stock stored of Product B at the regional warehouse to nine, which resulted in the number of kanbans stored of Product B dropping from three to two. A replenishment order was therefore generated by the regional warehouse at the Plant Finished Goods Store for five units of Product B, where the kanban size of Product B stored at Regional Warehouse #1 is five units. (Refer to table C.8 on page C-4.) Similarly Customer #3 ordered five units of Product C at Consignment Warehouse #1. The total number of stock stored of Product C at the consignment warehouse was reduced to 10, which reduced the number of kanbans from two to one. A replenishment order was generated by the consignment warehouse at Regional Warehouse #2 for 10 units of Product C. (Refer to table C.8 on page C-4.)

The replenishment order placed at Regional Warehouse #2 for Product C reduced the number of stock stored of Product C at the regional warehouse to five, thus the number of kanbans stored of Product C at the regional warehouse decreased from three to one. As a result a replenishment order was placed by Regional Warehouse #2 at the Plant Finished Goods Store for 10 units of Product C, where the 10 units resemble two kanbans. (Refer to table C.8 on page C-4.) This however initiated another replenishment order placed by the Plant Finished Goods Store for 10 units of Product C at the Plant, since the kanban number of Product C stored in the Finished Goods Store dropped by one.

The replenishment order placed at the Plant for 10 units of Product C by the Finished Goods Store caused the generation of replenishment orders for all three raw materials. Taking into

consideration the bill of materials (BOM) of Product C shown in Table C.1 on page C-2, 10 units of Product C are made from:

- 10 Units of Raw Material #1.
- 20 Units of Raw Material #2.
- 40 Units of Raw Material #3.

In taking the quantity required of each raw material and deducting it from the stock levels of the various raw materials, the number of kanbans consumed of each raw material was:

- One kanban of Raw Material #1.
- Two kanbans of Raw Material #2.
- Two kanbans of Raw Material #3.

On day 9 of the simulation run the replenishment order placed by Consignment Warehouse #1 for 10 units of Product C at Regional Warehouse #2 on day 8 was fulfilled. This correlated to the transport lead-time stated between Regional Warehouse #2 and Consignment Warehouse #1. (Refer to Table C.5. on page C-3.)

This process of verification was continued for 40 days in the simulation run, with the purpose of ensuring that the simulation model's working correlated to that of the concept model shown in figures 5.1 to 5.8 on pages 27 to 34. No errors were found and thus the assumption was made that the model is functioning correctly.

The verification of the simulation model's logic is however only part of the whole verification process. The verification process is continued using a stochastic supply chain model, and is discussed next.

7.1.2.2 Verification using a Stochastic Supply Chain Model

After verifying that the simulation model's logic adheres to that of the simulation concept model discussed in Chapter 5, a stochastic supply chain model is loaded in the simulation model (Refer to Appendix C). The stochastic model is used to verify the general working of the simulation model.

The *step* function is used in Simul8 to step through the simulation run, thus enabling verification of each order entity, the calculations surrounding the entity and data transferred to and from the database by each entity.

Apart from using the *step* function the simulation model is also run continuously for a number of times using various random number strings, with the purpose of flushing-out any errors that might occur during a simulation run. Furthermore part of the verification process is verifying the time it takes to complete a simulation run; since this has an enormous impact on the time it takes the optimisation tool in reaching an optimum solution. The next section covers the modifications made to the simulation model in order to improve its run time speed.

7.1.2.3 Optimisation of the Simulation Model

The simulation model is used in conjunction with the Harmony Search Algorithm (HSA) in the optimisation process, as discussed in Chapter 6. This implies that the simulation model must run a considerable number of times before the optimum solution is determined (Refer to figure 6.3 on page 60.) and thus has a big influence on the optimisation process's speed.

In order to improve the speed at which the simulation model completes a simulation run the various tables accessed by the model from the database are temporarily stored in Simul8 at the beginning of each run. Consequently the database is accessed only at the beginning and the end of each simulation run, which has a time saving implication; since the time it takes to read a value from the database is considerably more when comparing it to the time it takes to read a variable stored in Simul8. This is due to the database being stored on a computer's hard drive while the data stored in Simul8 is stored in a computer's memory. The tables temporarily stored in Simul8 are: *rsCustomers*, *rsCustomers_Products*, *rsProducts_Locations*, *rsRawMat_Locations* and *rsProducts_RawMat*.

To access the data stored in Simul8 each table in Simul8 is assigned with a *counter*, *pointer* and a *binary search procedure*. The *counter* simply states the number of entries in a table and is used by its corresponding *binary search procedure*. Conversely the *pointer* states the number of a specific entry in the table and is the output of its corresponding *binary search procedure*, where the *binary search procedure* is responsible for finding a specific entry

stored in its table. The various *binary search procedures* are shown in table 7.1, stating the purpose and showing the input variables for each of the procedures.

Table 7.1: The binary search procedures with their respective input variables.

<i>Binary Search Procedure</i>	<i>Purpose</i>
rsCustomers_Products(Customer_ID)	Sets pointer to the first entry of the customer-product combination for the supplied customer ID. (The products are sorted according to their product IDs for each customer.)
rsProducts_Locations(Product_ID, Location ID)	Sets pointer to specified product-location combination.
RawMat_Locations(RawMat_ID, Location_ID)	Sets pointer to specified raw material-location combination.
rsProducts_RawMat(Product_ID, RawMat_ID)	Sets pointer to specified product-raw material combination.

The simulation model thus never communicates with the database during a simulation run, but rather calls the appropriate *binary search procedure*. The procedure uses the input variables and finds the corresponding entry in the table stored in Simul8. The position number of that entry is then assigned by the procedure to its corresponding *pointer*. The *pointer* then enables the simulation model to access the relevant data in its table without having to search through the whole table.

The modification made to the simulation model increased the simulation run time speed approximately 30 times. The verification process described in 7.1.2.1 and 7.1.2.2 is again followed after the improvements were made, thus ensuring that the simulation model is still functioning according to the concept model discussed in 5.2. (Refer to figures 5.1 to 5.8 on pages 27 to 34.)

7.1.3 Optimisation Algorithm

The implementation of the optimisation algorithm needs to be verified against the working of the Harmony Search Algorithm (HSA), as discussed in Chapter 6. (Refer to figure 6.3 on page 60.) The implementation of the HSA in the optimisation tool can be divided into five subcomponents, namely:

1. The generation of the initial Harmony Memory (HM).
2. The generation of a random inventory solution.
3. The storing of a new random inventory solution in the database and prompting the simulation model to start a new simulation.
4. Retrieving the results from a finished simulation run and determining the value of the proposed inventory solution.
5. Evaluating the proposed solution, this includes:
 - 5.1 Determining if the proposed solution is valid.
 - 5.2 Updating the HM if necessary.
 - 5.3 Determining whether the optimisation process needs to be terminated or not.
 - 5.4 Is the report generator activated when the optimisation process is complete?

Each of the above mentioned components is verified by stepping through the program code and comparing the data stored in the database to what is expected, the process is similar to the process followed when verifying the simulation model.

7.1.4 Report Generator

The verification of the report generator is very trivial and only implies the verification of the exporting of the correct inventory solution values from the database to Microsoft Excel in the correct format.

In the following section the methods used to validate the tool is discussed. The validation of the tool is concerned with the inventory solution proposed by the tool for a specific problem and comparing it to the actual optimum inventory solution for the problem, i.e. the validation of the tool gives an indication as to how good the tool performs.

7.2 VALIDATION OF THE OPTIMISATION TOOL

The validation of the supply chain optimisation tool is important, since it gives an indication of how good the inventory solution proposed by the tool is. The validation of the tool merely entails the validation of the implementation of the Harmony Search Algorithm (HSA), since it is the HSA that governs the whole optimisation process.

In order to validate the tool a small supply chain model was built for evaluation. Refer to Appendix E for a detailed description of the model. The model in brief consists of a regional warehouse and a plant, where the plant has its respective raw material and finished goods warehouses. There are two products and two raw materials stored in the various locations defined in the supply chain, while two customers place orders for the products at either one of the plant or the regional warehouses.

To validate the HSA's result it needs to be compared against the actual optimum solution. The actual optimum solution can however only be determined in an extensive search, where all the various inventory combinations are tested against each other. Table 7.2 shows the products and raw materials stored in the various locations in the supply chain, their respective kanban size- and kanban number ranges and the number of locations they are stored at in the supply chain. The number of combinations that needs to be tested (n) is determined as follows:

$$n = (K_S K_N)^M \quad \dots (7.1)$$

Where:

K_S – The range of the kanban size.

K_N – The range of the kanban number.

M – The sum of all the products and raw materials locations.

Table 7.2: Validation values for the optimisation tool.

<i>Product / Raw Material</i>	<i>Number of Locations Stored At</i>	<i>Range of:</i>	
		<i>Kanban Size</i>	<i>Kanban Number</i>
Raw Material #1	1	2	3
Raw Material #2	1	2	3
Product #1	1	2	3
Product #2	2	2	3

Applying equation 7.1 to the defined supply chain, the number of inventory combinations is as follows:

$$\begin{aligned}
 n &= (K_S K_N)^M \\
 &= (2 \times 3)^5 \\
 &= 7,776
 \end{aligned}
 \tag{7.2}$$

The results of the exhaustive search, where all the 7,776 combinations are tested, are shown in table 7.3, against the results obtained from three different HSA optimisations. The Customer Service Level (CSL) requirement for all four tests are set at 95%, *this implies that only the inventory solutions whose CSL is 95% or higher will be considered*. The returned inventory levels thus guarantee a minimum CSL of 95%. (Refer to equation 3.10 on page 17.) The Termination Criteria for all three of the HSA optimisations are set at 50 iterations, implying that if no improvement is made after 50 iterations the HSA is terminated. All of the testing for both the exhaustive search and the three HSA optimisations are done on the same computer, where table 7.4 shows the computer's specifications.

Table 7.3: Comparison of validation results for the various scenarios.

		<i>Exhaustive Search</i>	<i>Harmony Search Algorithm</i>		
			<i>Run #1</i>	<i>Run #2</i>	<i>Run #3</i>
Optimisation Speed [in minutes]		542	129	12	31
Objective Function's Value		2,642.60	2,849.80	2,643.60	2,642.60
Raw Material 1	KS	30	30	30	30
	KN	2	2	2	2
Raw Material 2	KS	30	20	30	30
	KN	2	2	2	2
Product 1 (Plant WH)	KS	5	5	5	5
	KN	2	2	2	2
Product 2 (Plant WH)	KS	10	10	10	10
	KN	3	3	3	3
Product 2 (Regional WH)	KS	5	5	5	5
	KN	2	2	2	2

* KS – Kanban Size

** KN – Kanban Number

Table 7.4: The specifications of the computer used to verify and validate the supply chain optimisation tool.

Computer Model	HP Compaq NX9010
Processor	Pentium 4 CPU 2.66 GHz
Memory	198 MB
Operating System	Windows XP Professional

From table 7.3 it can be seen that two of the three HSA optimisation runs propose the same inventory solution than that of the exhaustive search, while the other solution only differs in the kanban size proposed for Raw Material 2. The development of the supply chain optimisation tool is concluded in the next chapter, where the various conclusions drawn from the thesis are stated together with recommendations on further aspects to be studied.

CHAPTER 8

CONCLUSION & RECOMMENDATIONS

As stated in Chapter 1, the thesis is aimed at developing a generic Just-In-Time (JIT) supply chain optimisation tool, which will enable an organisation to determine the optimum stock levels throughout its supply chain. The optimum stock levels will ensure that a predefined customer service level is maintained at the minimum cost to the company. It can be concluded from the results stated in Chapter 7, table 7.3 on page 72, that the solutions obtained for the tool is at least very close to optimal considering the given benchmark problem.

The Harmony Search Algorithm (HSA) is responsible for governing the optimisation process, and since the results obtained are excellent one can conclude that the HSA is a metaheuristic algorithm capable of reaching optimum or near-optimum solutions fairly easily. It is however not possible to determine exactly when the HSA is converging to a solution, as shown in figure 6.2 on page 59. Thus the implementation of other metaheuristic algorithms, such as the Genetic Algorithm (GA) or the Population-Based Incremental Learning (PBIL) algorithm is recommended for future study. The implementation of more than one algorithm also gives the optimiser the advantage of using more than one algorithm's results when suggesting an inventory solution.

In order to develop the optimisation tool various assumptions are made, as stated in chapters 3 to 5, these assumptions raise the question as to whether the tool is generic for all JIT supply chains. At best one can say that the optimisation tool is generic for a subset of the whole JIT supply chain spectrum, where the subset adheres to the assumptions made in the aforementioned chapters. The payoff between a simulation-optimisation study for a specific organisation compared to using the generic optimisation tool is also apparent. It is recommended that the development of other simulation concept models together with their respective objective functions be researched for future study. This will enable the tool to become more generic, since the user will be able to select the simulation model and objective function most suitable to his organisation.

It can be concluded that the supply chain optimisation tool as currently developed will give a reasonably good inventory solution for any organisation, bearing in mind the assumptions made. There are especially three assumptions that can have a dramatic influence on the validity of the inventory solution proposed, these assumptions are:

1. The storage facility's required size will be dependant on the maximum inventory size, as determined by the defined kanbans for the various products stored in the facility. Thus the fact that part of the inventory will always be on-route to the facility is ignored. (Refer to 3.3.2.2, p.14)
2. There is no capacity constraint in any of the plants defined within the supply chain. (Refer to 5.2, p.35)
3. The lead-times in the organisation's supply chain are short enough to ensure that batches of 365 days are not correlated and thus statistically independent. (Refer to 5.5.1, p. 48)

If an organisation's storage cost is substantial, the first assumption stated above will dramatically skew the result obtained for the objective function after each simulation run. Although the storage cost is set up to be conservative, thus catering for the worst-case-scenario, it might overshadow the transport cost and thus give the wrong kanban size, kanban number ratio for the products and raw materials stored in the supply chain.

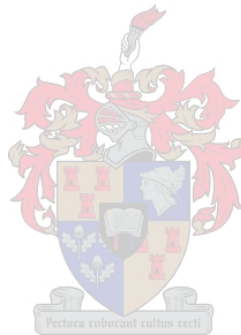
The second assumption is also very important to take note of, since in reality if there is a significant capacity constraint in an organisation's plants it will increase the amount of stock carried in the supply chain downstream from them. The manufacturing distribution can however be adjusted to cater for delays in production within the optimisation tool.

The most critical assumption of all is however the last assumption mentioned above, if an organisation has long lead-times, then the batches of 365 days will not be big enough. This will imply that the output parameters of the simulation model are batched in such a way that they are statistically dependent, which will result in the objective function's value to be deemed worthless. Thus a recommendation for future study is the automation of the batch means approach [Bekker 2003, p. 52] and the implementation thereof within the simulation model, therefore ensuring that the batches are statistically independent.

REFERENCES

- Albertyn, M. & Kruger, P.S. 2003. Genetic Building Blocks for Simulation Modelling of Stochastic Continuous Systems. *South African Journal of Industrial Engineering*, 14(2):47-61.
- Bekker, J. 2003. Simulation of discrete-event stochastic processes. Unpublished course notes (Post-graduate course in simulation). Stellenbosch: Department of Industrial Engineering, Stellenbosch University.
- Bekker, J. 2004a. Metaheuristics for Simulation Optimisation in Manufacturing. *International Conference on Competitive Manufacturing*. February 2004, Stellenbosch:407-412.
- Bekker, J. 2004b. Some meta-heuristics for combinatorial optimisation. Unpublished course notes (Post-graduate course in modelling). Stellenbosch: Department of Industrial Engineering, Stellenbosch University.
- Chase, R.B., Aquilano, N.J. & Jacobs, F.R. 2001. *Operations management for competitive advantage 9th edition*. New York: McGraw Hill/Irwin.
- Devore, J.L. & Farnum, N.R. 1999. *Applied statistics for engineers and scientists*. Pacific Grove, CA: Duxbury.
- Dobler, D.W. & Burt, D.N. 1996. *Purchasing and supply management: text and cases 6th Edition*. New York: McGraw-Hill.
- Franzese, L.A.G., Fioroni, M.M. & Botter, R.C. 2003. Rail road simulator on closed loop. *Proceedings of the 2003 Winter Simulation Conference*. December 2003, New Orleans, LA:1602-1606.
- Geen, Z.W., Kim, H.K. & Loganathan, G.V. 2001. A New Heuristic Optimization Algorithm: Harmony Search, *Simulation*, 76(2), February:60-68.
- Hamoen, S.C. & Moens, D.J. 2002. Logistic simulator for steel production factories. *Proceedings of the 2002 Winter Simulation Conference*. December 2002, San Diego, CA:1315-1318.
- ITtoolbox Supply Chain Knowledge Base* [In time]. 2004. Available: <http://supplychain.ittoolbox.com>. [2004, 21 November].
- Jain, S. & Choong, N.F. 2002. Modelling computer assembly operations for supply chain integration. *Proceedings of the 2002 Winter Simulation Conference*. December 2002, San Diego, CA:1165-1173.

- Rossetti, M.D. & Chan, H.T. 2003. A prototype object-oriented supply chain simulation framework. *Proceedings of the 2003 Winter Simulation Conference*. December 2003, New Orleans, LA:1612-1620.
- Shapiro, J.F. 2001. *Modeling the supply chain*. Pacific Grove, CA: Duxbury Thomson Learning.
- Silver, E.A., Pyke, D.F. & Peterson, R. 1998. *Inventory management and production planning and scheduling*. New York: John Wiley & Sons.
- Stchedroff, N. & Cheng, R.C.H. 2003. Modelling a continuous process with discrete simulation techniques and its application to LNG supply chains. *Proceedings of the 2003 Winter Simulation Conference*. December 2003, New Orleans, LA:1607-1611.



BIBLIOGRAPHY

- Albertyn, M. & Kruger, P.S. 2003. Genetic Building Blocks for Simulation Modelling of Stochastic Continuous Systems. *South African Journal of Industrial Engineering*, 14(2):47-61.
- Ansari, A. & Modarress, B. 1990. *Just-In-Time Purchasing*. New York: The Free Press.
- Bekker, J. 2000. Simulation in Supply Chains: An Arena basis. *South African Journal of Industrial Engineering*, 11(1):1-15.
- Bekker, J. 2003. Simulation of discrete-event stochastic processes. Unpublished course notes (Post-graduate course in simulation). Stellenbosch: Department of Industrial Engineering, Stellenbosch University.
- Bekker, J. 2004a. Metaheuristics for Simulation Optimisation in Manufacturing. *International Conference on Competitive Manufacturing*. February 2004, Stellenbosch:407-412.
- Bekker, J. 2004b. Some meta-heuristics for combinatorial optimisation. Unpublished course notes (Post-graduate course in modelling). Stellenbosch: Department of Industrial Engineering, Stellenbosch University.
- Chase, R.B., Aquilano, N.J. & Jacobs, F.R. 2001. *Operations management for competitive advantage 9th edition*. New York: McGraw Hill/Irwin.
- Dalal, M.A., Bell, H. & Denzien, M. 2003. Initializing a distribution supply chain with live data. *Proceedings of the 2003 Winter Simulation Conference*. December 2003, New Orleans, LA:1621-1626.
- Devore, J.L. & Farnum, N.R. 1999. *Applied statistics for engineers and scientists*. Pacific Grove: Duxbury.
- Dobler, D.W. & Burt, D.N. 1996. *Purchasing and supply management: text and cases 6th Edition*. New York: McGraw-Hill.
- Dyckhoff, H. (ed), Lackes, R. (ed) & Reese, J. (ed), 2004. *Supply Chain Management and Reverse Logistics*. Berlin: Springer.
- Franzese, L.A.G., Fioroni, M.M. & Botter, R.C. 2003. Rail road simulator on closed loop. *Proceedings of the 2003 Winter Simulation Conference*. December 2003, New Orleans, LA:1602-1606.
- Fredendall, L.D. & Hill, E. 2001. *Basics of Supply Chain Management*. Boca Raton, Florida: St. Lucie Press.

- Geen, Z.W., Kim, H.K. & Loganathan, G.V. 2001. A New Heuristic Optimization Algorithm: Harmony Search, *Simulation*, 76(2), February:60-68.
- Hamoen, S.C. & Moens, D.J. 2002. Logistic simulator for steel production factories. *Proceedings of the 2002 Winter Simulation Conference*. December 2002, San Diego, CA:1315-1318.
- Handfield, R.B. & Nichols Jr., R. L. 1999. *Introduction to Supply Chain Management*. Upper Saddle River, New Jersey: Prentice Hall.
- ITtoolbox Supply Chain Knowledge Base* [In time]. 2004. Available: <http://supplychain.ittoolbox.com>. [2004, 21 November].
- Jain, S. & Choong, N.F. 2002. Modelling computer assembly operations for supply chain integration. *Proceedings of the 2002 Winter Simulation Conference*. December 2002, San Diego, CA:1165-1173.
- Kelton, W.D., Sadowski, R.P. & Sadowski, D.A. 2002. *Simulation with Arena*. New York: MacGraw Hill.
- Kendel, K.E. & Kendel, J.E. 2002. *System analysis and design 5th edition*. Upper Saddle River, New Jersey: Prentice Hall.
- Rayward-Smith, V.J., Osman, I.H., Reeves, C.R. & Smith, G.D. 1996. *Modern Heuristic Search Methods*. New York: John Wiley & Sons.
- Rossetti, M.D. & Chan, H.T. 2003. A prototype object-oriented supply chain simulation framework. *Proceedings of the 2003 Winter Simulation Conference*. December 2003, New Orleans, LA:1612-1620.
- Shapiro, J.F. 2001. *Modeling the supply chain*. Pacific Grove, CA: Duxbury Thomson Learning.
- Silver, E.A., Pyke, D.F. & Peterson, R. 1998. *Inventory management and production planning and scheduling*. New York: John Wiley & Sons.
- Simul8 Corporation. 1999. *Simul8 Manual and Simulation Guide*. Herndon, V.A.
- Stchedroff, N. & Cheng, R.C.H. 2003. Modelling a continuous process with discrete simulation techniques and its application to LNG supply chains. *Proceedings of the 2003 Winter Simulation Conference*. December 2003, New Orleans, LA:1607-1611.
- Yang, J. & Pan, J.C. 2004. Just-In-Time Purchasing: an integrated inventory model involving deterministic variable lead time and quality improvement investment. *International Journal of Production Research*, 42(5):853-863.

APPENDIX A

THE USER'S GUIDE TO THE OPTIMISATION TOOL

A.1 INSTALLATION OF SOFTWARE TOOL

- Install Simul8 Release 10, the Professional Edition.
- Create a Folder on the Hard Drive where the Optimiser can be installed.
- Copy *JIT Supply Chain Optimiser.exe* and *Supply Chain Data.mdb* to the newly created folder.

A.2 CREATE AN ODBC DATA SOURCE FOR THE OPTIMISER'S DATA

- Run the ODBC Data Source Administrator (Control Panel → Administrative Tools → Data Sources (ODBC)).
- Add a new ODBC data source, refer to figure A.1.
 - Select the Microsoft Access Driver (*.mdb) and click the “Finish” button.
 - The Data Source Name should be: *SCData*.
 - Select the *Supply Chain Data.mdb* database in the folder where it has been copied to and then click the “OK” button.

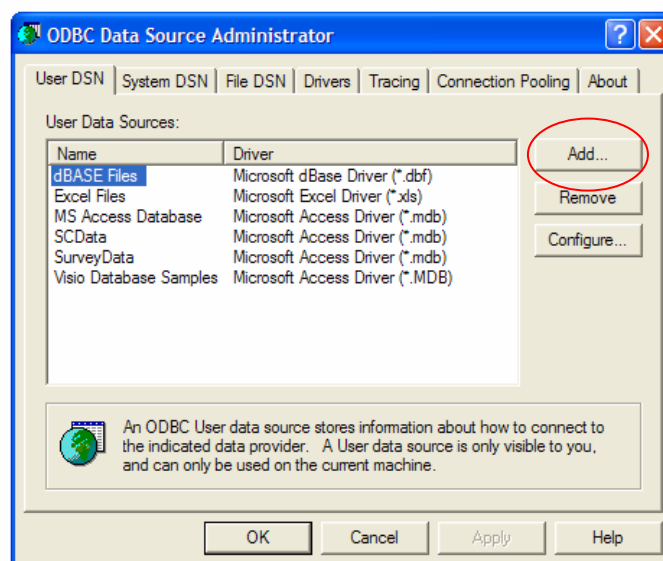


Figure A.1: Adding a new ODBC data source.

A.3 DEFINING A SUPPLY CHAIN IN THE OPTIMISER

- Load the Supply chain optimiser by double-clicking the *JIT Supply Chain Optimiser.exe* file.
- Click on the *Model Builder* option to load the model builder, refer to figure A.2.
- The raw materials, products, plants, regional warehouses, consignment warehouses and customers currently loaded on the system, can be seen by selecting the appropriate option, refer to figure A.3.
- When one of the abovementioned entities are displayed, the user has the option of adding, deleting or editing an entity, refer to figure A.4.

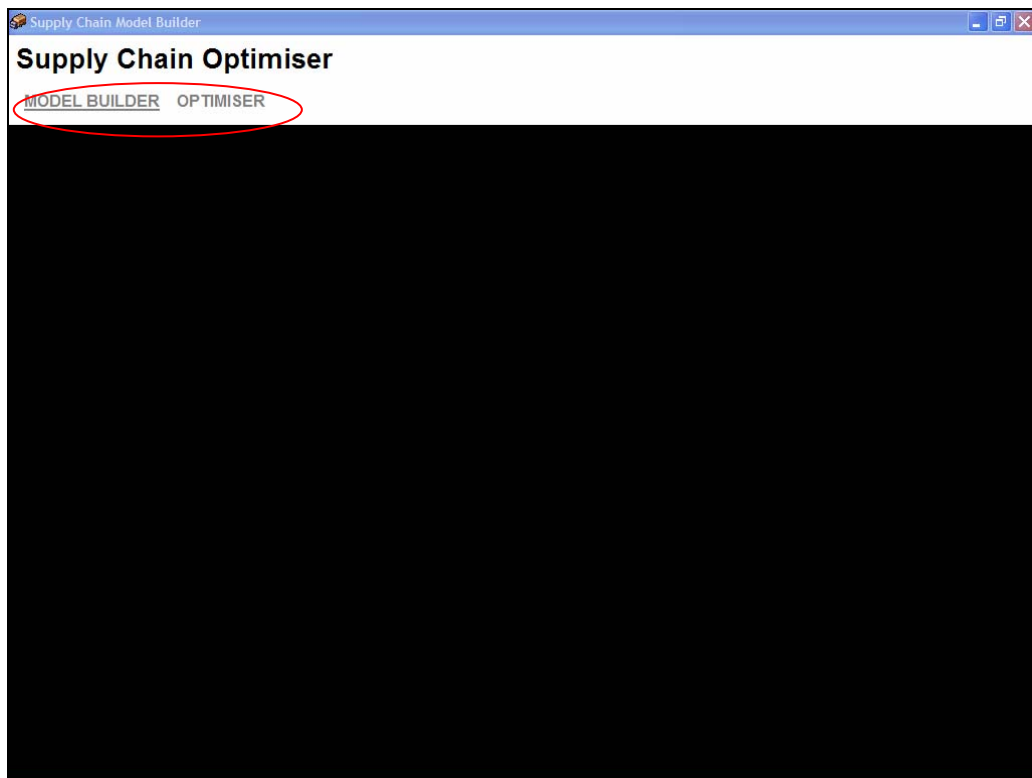


Figure A.2: The user can toggle between the supply chain model builder and the optimiser by selecting the appropriate option.

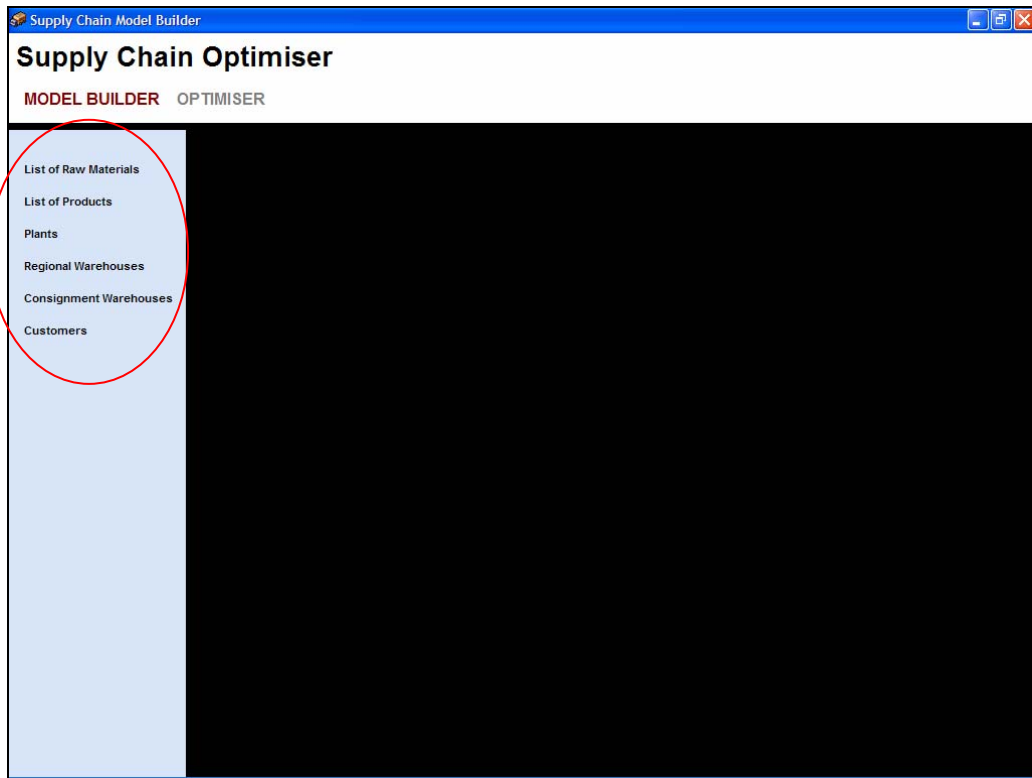


Figure A.3: The various entities defined in the model builder.

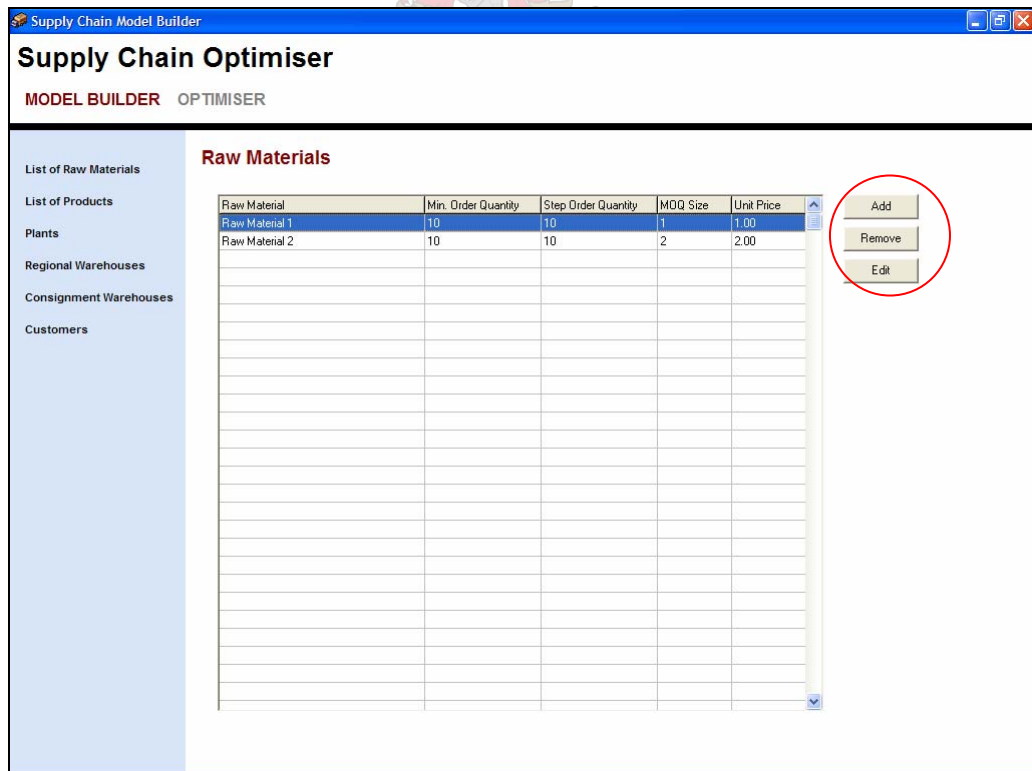


Figure A.4: The user can add, delete or edit an entity.

A.3 DETERMINING THE OPTIMUM INVENTORY SOLUTION FOR A DEFINED SUPPLY CHAIN

- Load the optimiser, refer to figure A.2.
- The optimiser has three options, namely: Define the solution space, the setup of the optimiser and the running of the optimiser. (Refer to figure A.5)
- In defining the solution space the following are set:
 - *Maximum Number of Kanbans*: This defines the boundary for the number of kanban's generated by the optimisation algorithm.
 - *Maximum Kanban "Size"*: The kanban size of a product is determined by the minimum order quantity and step order quantity defined for each specific product in every location. If the *Maximum Kanban "Size"* value is set to 1, only the minimum order quantity will be taken into account by the optimisation algorithm. If the value assigned is n , where n is bigger than 1, then a kanban size quantity can be set by the optimisation algorithm from the minimum order quantity, to the minimum order quantity + $(n-1) \times$ step order quantity.
 - *Customer Service Level*
 - *Capital Interest Rate*
- In the optimiser's setup the following is set:
 - *Best Solutions Memory Size*: This is the Harmony Search Algorithm's (HSA) Harmony Memory (HM) size.
 - *Tolerance Run Length*: The number of iterations that needs to pass, since the last improvement was made, before the optimiser algorithm stops and declares its optimum solution.
 - *Best Solution Considering Rate*: This is the HSA's Harmony Memory Considering Rate parameter.
 - *Pitch Adjustment Rate*: The HSA's Pitch Adjustment Rate parameter.
- Run the optimiser as soon as the solution space and optimiser are set up correctly, figure A.6 shows the interface of the optimiser when it is busy running.
- The inventory solution proposed by the optimiser is exported to Microsoft Excel when the optimisation process is terminated, refer to figure A.7.

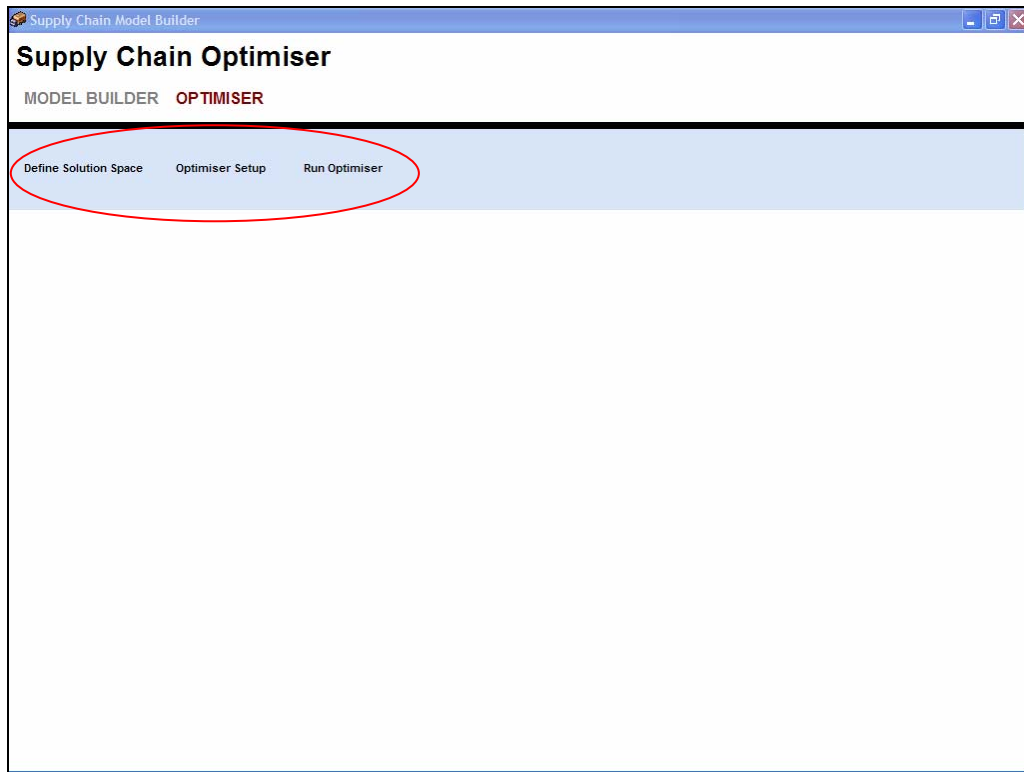


Figure A.5: The optimiser's interface contains three options, namely: Define Solution Space, Optimiser Setup and Run Optimiser.

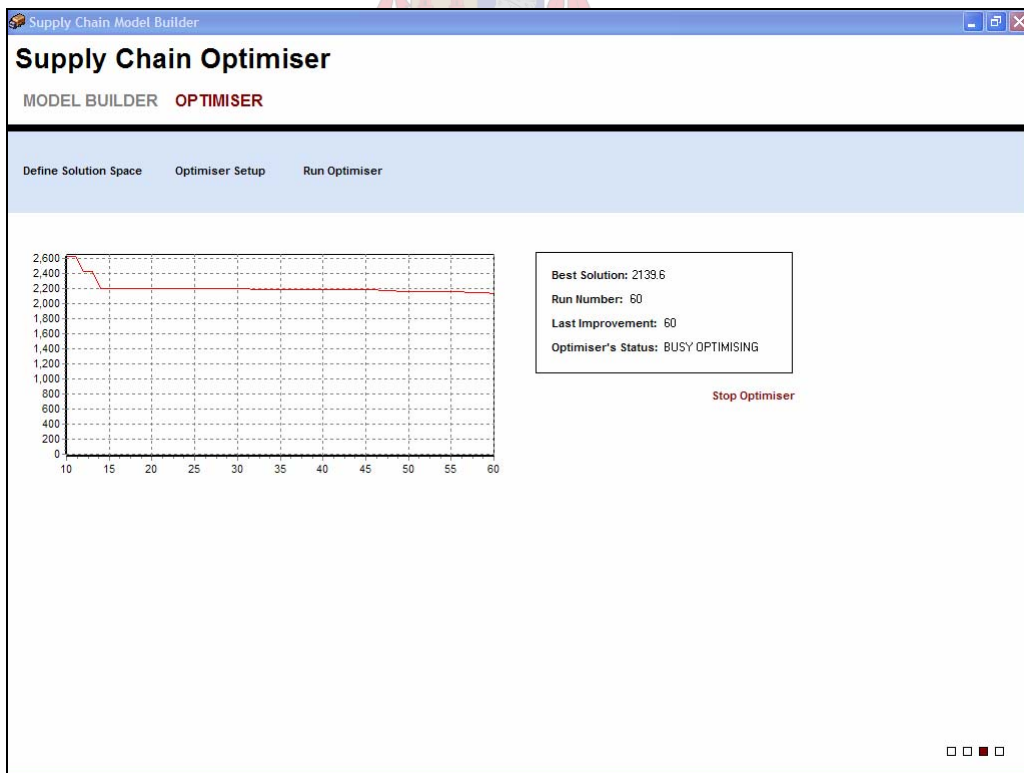


Figure A.6: The running of the supply chain optimiser.

Raw Material		Plant
Raw Material 1	Kanban Size	150
	Kanban Number	10
Raw Material 2	Kanban Size	170
	Kanban Number	7
Raw Material 3	Kanban Size	6
	Kanban Number	8
Raw Material 4	Kanban Size	12
	Kanban Number	2
Raw Material 5	Kanban Size	17
	Kanban Number	4

Figure A.7: The output of the supply chain optimiser.



APPENDIX B

THE DATABASE'S DATA DICTIONARY

Table B-1: The database's data dictionary.

<i>Name</i>	<i>Type</i>	<i>Purpose</i>
rsLocationTypes	Table	Stores the various location types defined in the supply chain, e.g. Consignment Warehouse.
LocationType_ID	Integer	Primary key.
LocationType	String	Location type's name.
rsLocations	Table	Stores the various locations defined in the supply chain.
Location_ID	Integer	Primary key.
Location_Name	String	The location's name.
LocationType_ID	Integer	The location type's key as a foreign key.
StorageCost	Decimal	The cubic meter storage cost at the location.
rsRawMat	Table	Stores all the raw materials required to manufacture products.
RawMat_ID	Integer	Primary key.
RawMat_Name	String	The raw material's name.
UnitCost	Decimal	The unit cost quoted by the supplier.
MinOrderQuantity	Integer	The minimum quantity allowed to order from the supplier.
StepOrderQuantity	Integer	The quantity by which an order can incrementally be increased by.
MOQ_Size	Decimal	The size in cubic meters of the minimum order quantity.
rsProducts	Table	Stores the entire product range stored throughout the defined supply chain.
Product_ID	Integer	Primary key.

Table B-1 (continued): The database's data dictionary.

<i>Name</i>	<i>Type</i>	<i>Purpose</i>
Product_Name	String	The product's name.
rsProducts_RawMat	Table	Stores the bill of materials (BOM) of each product.
Product_ID	Integer	Primary key.
RawMat_ID	Integer	Primary key.
RawMatQuantity	Integer	The number of raw material units required to produce one unit of the product.
rsLocations_RawMat	Table	Stores the various raw materials stored in each location defined in the supply chain.
Location_ID	Integer	Primary key.
RawMat_ID	Integer	Primary key.
TransCost	Decimal	The cost involved in transporting one unit of raw material from the supplier to the storage location.
OrderCost	Decimal	The cost involved in placing a replenishment order, irrespective of the quantity ordered.
AvgLeadTime	Decimal	The average lead-time to replenish the raw material. (Normal Distributed)
StdDevLeadTime	Decimal	The standard deviation of replenishing the raw material. (Normal Distributed)
KanbanSize	Integer	An integer value representing the kanban size stored of the raw material, where: <ul style="list-style-type: none"> • 0 = Minimum order quantity • 1 = Minimum order quantity + (1 × step order quantity) • 2 = Minimum order quantity + (2 × step order quantity) • Etc.
KanbanNumber	Integer	The number of kanban's stored of the raw material at any given time; the minimum value will always be 2.

Table B-1 (continued): The database's data dictionary.

<i>Name</i>	<i>Type</i>	<i>Purpose</i>
Stock	Decimal	The total number of stock available of the raw material at any given time.
OrigKanbanNumber	Integer	The original kanban number proposed by the optimisation algorithm before a simulation run.
rsLocations_Products	Table	Stores the various products stored in each location in the defined supply chain.
Location_ID	Integer	Primary key.
Product_ID	Integer	Primary key.
ReplenishLocation_ID	Integer	A foreign key stating the location where to place a replenishment order for the product.
MakeToOrder	Boolean	Only applies to products stored at plants. If the field is set, the product is not stored in the finished goods warehouse.
MinOrderQuantity	Integer	The minimum order quantity allowed to be placed at the upstream replenishment location.
StepOrderQuantity	Integer	The quantity by which a replenishment order can incrementally be increased by.
CapitalValue	Decimal	The capital value of the product.
TransCost	Decimal	The per unit transportation cost from the replenishment location.
OrderCost	Decimal	The cost associated with generating a replenishment order, irrespective of the order quantity.
AvgLeadTime	Decimal	The average lead-time to transport the product from the replenishment location. (Normal Distributed)
StdDevLeadTime	Decimal	The standard deviation of the lead-time to transport the product to the replenishment location. (Normal Distributed)

Table B-1 (continued): The database's data dictionary.

<i>Name</i>	<i>Type</i>	<i>Purpose</i>
KanbanSize	Integer	An integer value representing the kanban size stored of the product, where: <ul style="list-style-type: none"> • 0 = Minimum order quantity • 1 = Minimum order quantity + (1 × step order quantity) • 2 = Minimum order quantity + (2 × step order quantity) • Etc.
KabanNumber	Integer	The number of kanbans stored of the product at any given time; the minimum value will always be 2.
Stock	Decimal	The total number of stock available of the product at any given time.
OrigKanbanNumber	Integer	The original kanban number proposed by the optimisation algorithm before a simulation run.
rsCustomers	Table	Stored all the customers defined in the supply chain.
Customer_ID	Integer	Primary key.
Customer_Name	String	The customer's name.
AvgArrival	Decimal	The inter-arrival rate of the customers. (Exponentially Distributed)
rsCustomers_Products	Table	Stores the various products ordered by each customer.
Customer_ID	Integer	Primary key.
Product_ID	Integer	Primary key.
Location_ID	Integer	The foreign key stating where the product is ordered by the customer.
ProductChance	Decimal	The chance that the customer will order this specific product when he arrives.
AvgQuantity	Decimal	The average quantity ordered of the product by the customers. (Normal Distributed)

Table B-1 (continued): The database's data dictionary.

<i>Name</i>	<i>Type</i>	<i>Purpose</i>
StdDevQuantity	Decimal	The standard deviation on the quantity ordered of the product by the customer. (Normal Distributed)
rsResults	Table	Stores the latest simulation run's results
Batch_ID	Integer	Primary key responsible for identifying the various batches in the simulation run's results.
GoodNr	Integer	The number of ordered units that could have been supplied directly to the customer in each batch.
TotalNr	Integer	The total number of units ordered in each batch.
TransCost	Decimal	The total transport cost incurred in each batch.
ReplenishCost	Decimal	The total replenishment cost incurred in each batch.
rsHarMem	Table	Stores the Harmony Memory during an optimisation exercise.
HarMem_ID	Integer	Primary key.
Score	Decimal	The total cost of the objective function used in the optimisation process.
rsHarMem_RawMat	Table	Stores the configuration of the raw materials for each harmony in the Harmony Memory
HarMem_ID	Integer	Primary key.
RawMat_ID	Integer	Primary key.
Location_ID	Integer	Foreign key stating the location where the raw material is stored.
KanbanNumber	Integer	The initial number of kanbans stored of the raw material, where the minimum value will always be 2.
KanbanSize	Integer	An integer value representing the initial kanban size stored of the raw material, where: <ul style="list-style-type: none"> • 0 = Minimum order quantity • 1 = Minimum order quantity + (1 × step order quantity) • 2 = Minimum order quantity + (2 × step order quantity)

Table B-1 (continued): The database's data dictionary.

<i>Name</i>	<i>Type</i>	<i>Purpose</i>
rsHarMem_Products	Table	Stores the configuration of all the products for each harmony in the Harmony Memory.
HarMem_ID	Integer	Primary key.
Product_ID	Integer	Primary key.
Location_ID	Integer	Foreign key stating the location where the product is stored.
KanbanNumber	Integer	The initial number of kanban's stored of the product, where the minimum value will always be 2.
KanbanSize	Integer	An integer value representing the initial kanban size stored of the product, where: <ul style="list-style-type: none"> • 0 = Minimum order quantity • 1 = Minimum order quantity + (1 × step order quantity) • 2 = Minimum order quantity + (2 × step order quantity) • Etc.

APPENDIX C

THE SUPPLY CHAIN MODELS USED TO VERIFY THE

WORKING OF THE SIMULATION MODEL

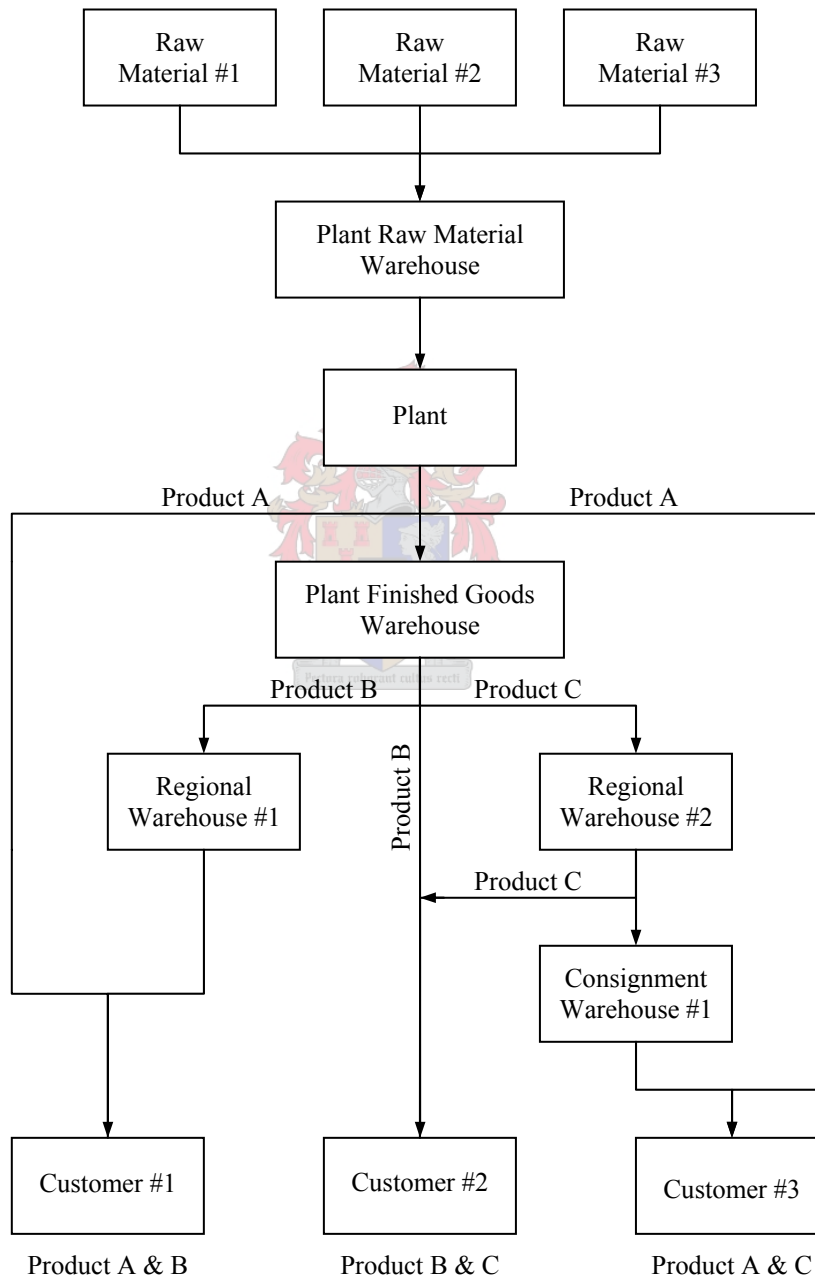


Figure C-1: The supply chain model used to verify the working of the simulation model.

Table C-1: The bill of materials (BOM) for the various products defined in both the deterministic and stochastic supply chain models.

	<i>Raw Material #1</i>	<i>Raw Material #2</i>	<i>Raw Material #3</i>
Product A	2	2	-
Product B	3	-	5
Product C	1	2	4

Table C-2: The average number of days between order placements for each customer in the two supply chain models.

	<i>Deterministic Model (Fixed Value)</i>	<i>Stochastic Model (Exponential Distribution)</i>
Customer #1	2	2
Customer #2	3	3
Customer #3	4	4

Table C-3: The order quantity distribution for each customer-product combination.

	<i>Deterministic Model (Fixed Value)</i>			<i>Stochastic Model's Average (Normal Distribution)</i>		
	<i>Product A</i>	<i>Product B</i>	<i>Product C</i>	<i>Product A</i>	<i>Product B</i>	<i>Product C</i>
Customer #1	3	3	-	3	3	-
Customer #2	-	2	2	-	2	2
Customer #3	5	-	5	5	-	5

* The standard deviation of the order quantity in the stochastic model is 1 unit for each product-customer combination.

Table C-4: The empirical distribution for the products ordered.

	<i>Product A</i>	<i>Product B</i>	<i>Product C</i>
Customer #1	50%	50%	-
Customer #2	-	50%	50%
Customer #3	50%	-	50%

Table C-5: The average number of days required to transport product between the various locations in the supply chain.

	<i>Deterministic Model</i> (Fixed Value)			<i>Stochastic Model's Average</i> (Normal Distribution)		
	Regional Warehouse #1	Regional Warehouse #2	Consignment Warehouse #1	Regional Warehouse #1	Regional Warehouse #2	Consignment Warehouse #1
Plant Finished Goods Warehouse	2	3	-	2	3	-
Regional Warehouse #1	-	-	-	-	-	-
Regional Warehouse #2	-	-	1	-	-	1
Consignment Warehouse #1	-	1	-	-	1	-

* The standard deviation of the transport time in the stochastic model is 1 day between each location in the supply chain.

Table C-6: The average number of days required to replenish the various raw materials ordered in the supply chain.

	<i>Deterministic Model</i> (Fixed Value)			<i>Stochastic Model's Average</i> (Normal Distribution)		
	Raw Material #1	Raw Material #2	Raw Material #3	Raw Material #1	Raw Material #2	Raw Material #3
Plant Raw Material Warehouse	5	3	6	5	3	6

* The standard deviation of the replenishment lead-time in the stochastic model is 1 day for each of the raw materials.

Table C-7: The average number of days required to manufacture the various product defined in the supply chain.

	<i>Deterministic Model</i> (Fixed Value)			<i>Stochastic Model's Average</i> (Normal Distribution)		
	Product A	Product B	Product C	Product A	Product B	Product C
Plant	3	3	5	3	3	5

* The standard deviation of the manufacturing lead-time in the stochastic model is 1 day for each product.

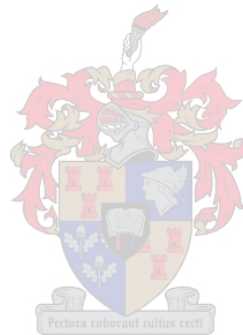
Table C-8: The kanban size and number of kanbans stored of each product at the various locations defined in the supply chain.

		<i>Plant Finished Goods Warehouse</i>	<i>Regional Warehouse #1</i>	<i>Regional Warehouse #2</i>	<i>Consignment Warehouse #1</i>
Product A	Kanban Number	-	-	-	-
	Kanban Size	-	-	-	-
Product B	Kanban Number	4	3	-	-
	Kanban Size	10	5	-	-
Product C	Kanban Number	4	-	3	2
	Kanban Size	10	-	5	10

Table C-9: The kanban size and number of kanbans stored of each raw material.

		<i>Plant Raw Materials Warehouse</i>
Raw Material #1	Kanban Number	4
	Kanban Size	15
Raw Material #2	Kanban Number	4
	Kanban Size	15
Raw Material #3	Kanban Number	4
	Kanban Size	15

APPENDIX D
THE SPREADSHEET USED IN VERIFYING THE
SIMULATION MODEL



APPENDIX E
THE SUPPLY CHAIN MODEL USED IN VALIDATING THE
SUPPLY CHAIN OPTIMISATION TOOL

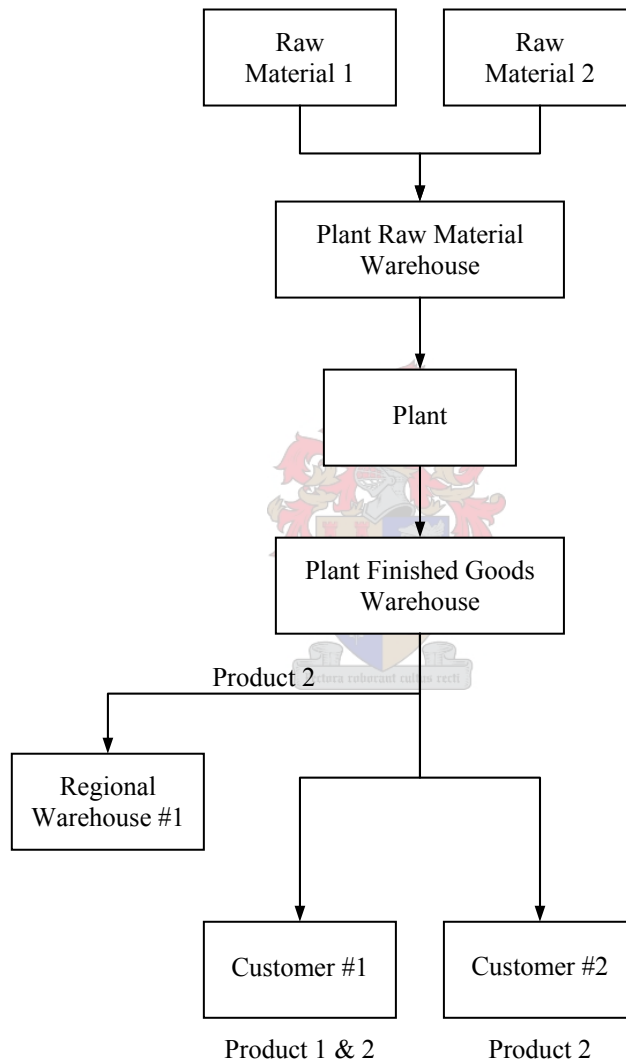


Figure E-1: The supply chain model used to validate the supply chain optimising tool.

Table E-1: The bill of materials (BOM) for the various products defined in the supply chain model.

	<i>Raw Material #1</i>	<i>Raw Material #2</i>
Product 1	1	0.5
Product 2	0.5	1

Table E-2: The average number of days between a customer's arrivals.

	<i>Average Number of Days Between Placing Orders</i>
Customer 1	3
Customer 2	5

Table E-3: The order quantity distribution for each customer-product combination.

	<i>Product 1</i>	<i>Product 2</i>
Customer 1	3	2
Customer 2	-	6

* The standard deviation of the order quantity is 1 unit for each product-customer combination.

Table E-4: The empirical distribution for the products ordered.

	<i>Product 1</i>	<i>Product 2</i>
Customer 1	75%	25%
Customer 2	-	100%

Table E-5: The average number of days required to replenish the various raw materials ordered in the supply chain.

	<i>Raw Material 1</i>		<i>Raw Material 2</i>	
	Average	Standard Deviation	Average	Standard Deviation
Plant Raw Material Warehouse	3	1	2	0.5

Table E-6: The average number of days required to manufacture the various product defined in the supply chain.

	<i>Product 1</i>		<i>Product 2</i>	
	Average	Standard Deviation	Average	Standard Deviation
Plant	3	0.5	3	1

