WAREHOUSE SIMULATION AND ANALYSIS

by

VICKY BEKKER
26065020

Submitted in partial fulfilment of the requirements for the degree of

BACHELORS OF INDUSTRIAL ENGINEERING

in the

FACULTY OF ENGINEERING, BUILT ENVIRONMENT AND INFORMATION TECHNOLOGY

UNIVERSITY OF PRETORIA

October 2010

EXECUTIVE SUMMARY

Nissan Rosslyn manufactures some Nissan models that are available in South Africa. They have identified a capacity problem in one of their part warehouses. The Excel based simulation model was constructed to be used as a decision tool in the evaluation of the specific warehouse as well as future analysis of different warehouses. The model's goal is to evaluate the parts flow of the warehouse in detail and identify problem areas. Different warehouse configurations are simulated in an attempt to find better alternatives. In this report the model is described, a few scenarios are simulated and future applications are investigated. The result of choosing a better alternative is a reduction in capacity problems and an overall more efficient warehouse.

Table of Contents

LIST OF FIGURES

1. INTRODUCTION

1.1 INTRODUCTION AND BACKROUND

Nissan is a world-wide motor vehicle company which has a wide range of vehicles starting with passenger cars to commercial vehicles. In the 1930's the Nissan Motor Company was registered after successfully completing the take-over of Kwaishinsha Automotive Company, which was first established in 1911. After the takeover, Nissan only grew, with facilities opening in the US by the 1980's and a manufacturing plant in 1984. With hardships falling on the motoring companies in the 1990's, alliances had to be formed in order for companies to survive. Nissan followed suite and formed an alliance with the French Renault group, which resulted in the fourth largest automotive company.

Nissan South Africa, which name changed from Automakers in 2001, supplies South Africa with vehicles from selected models in the Nissan range. Nissan SA makes up 8% of the automotive market. A manufacturing plant at Rosslyn near Pretoria, manufactures some of the models in the Nissan range.

The Rosslyn plant has various warehouses where the different parts for the vehicles are stored.

Nissan has identified a capacity problem in one of these warehouses. The warehouse in question has different sections and different bin types. Each part can only be allocated to certain section(s), and within that section(s), to certain bin type(s). Nissan uses random binning, where parts are not allocated to a specific bin. Some sections and bin types in the warehouse seem to have capacity problems and an analysis of the warehouse has to be performed to establish the problem areas.

1.2 PROJECT AIM

The aim of this project is to develop a dynamic warehouse configuration model which considers:

- projected changes in parts demand and
- changes in the strategic environment,

to be used as a decision tool in order to re-calculate and model the ideal warehouse configuration.

1.3 SCOPE

In the first stage of the project research concerning warehouse simulation and configuration will be conducted for a literature study. The warehouse in question will also be investigated to fully comprehend all factors influencing the inflow and outflow of parts.

A Microsoft Excel based program using Visual Basic Application will be written to simulate the warehouse operation. This model will simulate the inflow and outflow of materials. The flow of parts to sections, bin types within the section and specific bin within that bin type is also simulated. Bin capacities as well as policies concerning part shortage will be incorporated in the model to ensure that the model is as realistic as possible. The warehouse simulation will be comprehensive, simulating all flows within the warehouse. Flows outside the warehouse will be ignored as it has no importance in this specific project. Only parts flowing in and out of the warehouse will be considered.

Different warehouse configurations will be considered to evaluate its effect on capacity.

1.4 DOCUMENT STRUCTURE

In Chapter 2 a study will be done on relevant literature that is available. This will be followed by the model formation in Chapter 3. In this Chapter I discuss the ins-and-outs of the simulation model. After the completion of this discussion we will look at computational results in Chapter 4 followed by future work and conclusion in Chapter 5.

2. LITERATURE REVIEW

2.1 WAREHOUSE DESIGN

2.1.1 INTRODUCTION

It is not always possible to supply goods directly to the customer as it might not be cost effective. This is where a warehouse is appropriate. It results in the customer receiving goods within a short period of time. Warehouse design is so important when it comes to the efficiency of the warehouse. Even with this fact little literature exists that fully explain the approaches to warehouse design. Some examples are given below:

- ‘‘A search of the literature shows that very few papers deal with the general warehouse design problem” (Ashayeri and Gelders, 1985, p. 285);
- ‘‘In general, however, there is not a procedure for systematically analysing the requirement and designing a warehouse to meet the operational need using the most economic technology” (Rowley, 2000, p. 3);
- ‘‘A sound theoretical basis for a warehouse design methodology still seems to be lacking” (Rouwenhorst et al., 2000, p. 515);
- ‘‘A comprehensive and science-based methodology for the overall design of warehousing systems does not appear to exist” (Goetschalckx et al., 2002, p. 1).

Govindaraj et al. (2000) did research on warehouse designer and he concluded that they analyse the data using their intuition and experience. The designers then “make some initial design decisions”. It is then clear that there is a lack of a formal, streamlined approach. Some aspects of different parts of warehousing are explained further in this chapter.

2.1.2 WAREHOUSE CHARACTERISTICS

Rouwenhorst et al (1999) explains that a warehouse can be broken down into three elements namely processes, resources and organisation. These concepts are explained below:

- *Warehouse processes*

  The parts flow in the warehouse is divided into different processes.

1. *Receiving*

   Products arrive at the warehouse and are checked or repackaged.

2. *Storage*

   A storage area can be made up of two areas, namely a reserve area and a forward area. Reserve areas are occupied by bulk storage by using pallets racks. The forward area consists of racks.

3. *Order picking*

   Parts are retrieved from storage locations by using manual or partially automated methods.

4. *Shipping*

   At this phase in the process the products are checked, packed and removed from the warehouse.

- *Warehouse resources*

  Some examples of resources are given below:

  1. *Storage unit*

     Examples include pallets and boxes. Products are stored in/on these units.

  2. *Storage unit system*
  3. *Pick equipment*

     Parts can be retrieved by using pick equipment.

  4. *Material handling equipment*

     It includes equipment such as palletizers and sorter systems.

  5. *Personnel*

- *Warehouse organisation*

  This phase is concerned with the planning and control of the warehouse system. Warehouse processes need some organisational policies. These are explained as follows:

  1. *Assignment policy*

     Trucks are allocated to docks in this policy.

  2. *Storage policies*

     a. Dedicated:

        Parts are stored in specific predetermined areas. For example; a part is always stored at the same storage space.

b. Random:

   Operator decides where to store parts in this policy.

c. Class based:

   Parts are allocated to specific zones or sections in the warehouse.

d. Correlated:

   Parts are often used at the same time are placed in close proximity to each other.

3. *Forward/reserve and replenishment policies*

   Where parts are stored (forward or reserve) and the quantity of these products are determined in this policy. The replenishment is also determined.

4. *Zoning policies*

   The warehouse is divided into zones to be serviced by different order pickers. Two types exist i.e. parallel and sequential.

5. *Picking policies*

   Products can be picked individually or in batches. When products are picked in batches, these parts have to be sorted. The parts can be picked then sorted or sorted while picking.

6. *Routing policies*

   It is also necessary to determine the route and order in which products are picked. This can be achieved by this policy.

7. *Dock assignment policy*

   Parts and trucks are assigned to a dock.

8. *Operator and equipment assignment policies*

   Tasks are allocated to workers and equipment.

2.1.3 WAREHOUSE DESIGN STEPS

Baker and Canessa (2007) did a study on the literature available on the steps warehouse design companies use. Many methodologies throughout literature were evaluated and it was concluded that there were similarities. Most authors admitted the complexity of warehouse design and therefore tried to give a step by step approach. The steps explained are interrelated and thus adding to the complexity. Each step in the process has many possible solutions. This in turn may not make it possible to find an “optimal solution”.

The following steps are used by a certain company:

1. *Develop material flow*
2. *Evaluate relevant technology*
3. *Refine preferred option*
4. *Specify manning levels*
5. *Develop a warehouse layout:*

   Baker and Canessa (2007) describe s+ome methods used in developing a warehouse layout that were presented by other authors. These methods are explained below.

   - *Mulcahy (1994) suggests using one of the following:*
     - Block layout
     - Standard templates and layouts boards
     - Drawings (conventional or computer generated)
     - 3D modelling method
   - *Frazelle (2002b) uses some of the methods described above. He further explains a 5 step method as follows:*
     - Space requirement planning (block layout method): This method is used to determine the requirements for each section in the warehouse.
     - Material flow planning
     - Adjacency planning
     - Process location
     - Expansion planning
6. *Define process and system functionality*
7. *Develop a detailed design of warehouse*
8. *Simulation*
9. *Specify equipment*

2.1.4 TOOLS USED IN WAREHOUSE DESIGN

According to Baker & Canessa (2007) certain tools are used to complete each step in the warehouse design. Many tools are available to aid in each step. Tools can include some of the following:

- Spreadsheet models for data analyses, evaluating equipment types or equipment capacities

- Flow chart for determining material flow
- CAD (Computer-aided design) for developing warehouse layout
- Simulation models for warehouse evaluating or layout design

2.2 MODELLING

2.2.1. INTRODUCTION

A model can be defined as “a cut-down, simplified version of reality that captures the essential features of a problem and aids understanding.” (The Actuarial Education Company, 2010). This implies that a model must be simple while maintaining a sense of realism.

2.2.2. APPROACHES TO MODELLING

When a problem is identified, the following approaches can be followed (The Actuarial Education Company, 2010):

- Purchase a commercial model that is applicable to the problem
- Modify an existing model to suit the needs of the current project
- Develop a new model

Deciding which of these options to follow depends on the following factors:

- The required accuracy. This refers to how accurate the model has to reflect reality and how important it is for the model data to be precise
- Is relevant expertise available? Whether expertise is available depends on the employees that will interact with the model. This also refers to whether there is someone who is capable of developing a new model if necessary
- How many times will model be used?
- The desired flexibility of the model i.e. how easily the model is suited for other purposes
- The cost that is related to each option. The cost refers to development/ purchase cost as well as the cost of operation
- Whether the model (commercial or existing) is suited for the problem at hand

2.2.3. WHAT MAKES A GOOD MODEL

According to “The Actuarial Education Company” (2010) the requirements of a good model include the following:

- The model must be accurate for intended purpose, valid and sufficiently documented. It is necessary to be able to use the model in a wide variety of circumstances, while still being confident of its accuracy. Validity of the model refers to whether the model is appropriate to the real world circumstances of the problem. It is also necessary for all assumptions to be stated to make it easier for external users to understand.
- Relevant inputs must be appropriate to the business or situation being modelled.
- The model must be easily understood. The results must be communicated clearly to the user. Assumptions must be consistent throughout the model and appropriate to the circumstances.
- Outputs should be reasonable and be able to be verified. It should be possible to compare outputs to the real situation and determine whether those outputs are accurate.
- Over-complexity should be avoided so that interpreting outputs do not become too difficult and the model itself does not take too long to run. This can lead to the model becoming too costly. It must be remembered that not every single aspect of a situation can be modelled.
- It should be possible to change and refine the model to improve it.

2.2.4. DETERMINISTIC AND STOCHASTIC MODELS

It is necessary to decide between using a deterministic model, stochastic model or a combination of the two. The difference, advantages and disadvantages are explained as follows (The Actuarial Education Company, 2010):

- Deterministic model:
  Definition: A model where outcomes are not dependant on variation, but rather on precise data. The result will be that the same output is generated for the same input. (Business dictionary, 2010)

Advantages:

- Easily explained to non-technical personnel
- Quicker to run and usually easier to design

Disadvantages:

- It is difficult to predict which scenarios should be tested. When raw data and deterministic assumptions are used in a model only that scenario is modelled. Using this approach can lead to certain scenarios, that can negatively affect the business, not being identified.

- Stochastic model:

  Definition: A model which uses probability distribution to model a range of values for each variable.

  Advantages:

  - A wider range of scenarios can be modelled
  - The quality of the result is better than in deterministic modelling
  - Results in more information for the end users. With more scenarios being modelled, greater information about the situation is gathered.

  Disadvantages:

  - The model will take longer to run
  - The model is more complex
  - Accuracy depends on the correctness of the distribution function used

A combination of the two models can also be used.

The following steps can be followed when developing a deterministic or stochastic model: (The Actuarial Education Company, 2010)

1. Specify the purpose of the model.
2. Collect and modify data where applicable.
3. *Deterministic:* Identify variables and parameters

   Stochastic: Choose a density functions for each of the variables and describe relevant correlation between them.
4. Assign values to parameters and variables (if it is not being modeled stochastically).
5. Construct a model
6. Check that goodness of fit is at an acceptable level. To achieve this previous year's data is run and then compared to the model.

7. Use a different fit if the original fit is not acceptable.
8. *Deterministic:* Run the model using actual values for the variables followed by using estimated values. In order to determine the sensitivity of the parameters the model must be run several times.
   Stochastic: Run model several times using random values of the distribution function and compile the results.

2.2.5. STATIC AND DYNAMIC MODELS

Time is the distinguishing factor between the two models. Dynamic models consider time whereas static models do not. (Kelton, Sadowski & Sturrock, 2007)

2.2.6. CONTINUOUS AND DISCRETE MODELS

The difference between the two models is concerned with when the state in a system can change. Continuous models allow continuous changes, while discrete models only allow changes at specific instances of time. A model with both these features also exists. (Kelton, Sadowski & Sturrock, 2007)

2.3 SIMULATION AND PROGRAMMING TOOLS

Various simulation and programming tools are commercially available to aid in developing a warehousing model. Some of these tools include ProModel, Arena, Matlab and Excel. Some of these tools are made specifically for simulating aspects of warehouses, whereas others can be adapted to achieve the necessary result.

2.3.1. PROMODEL

ProModel is a commercially available software tool. It can be used for manufacturing, logistics and warehousing applications. The following figure shows an example of a warehouse model using ProModel (ProModel Products, 2010):

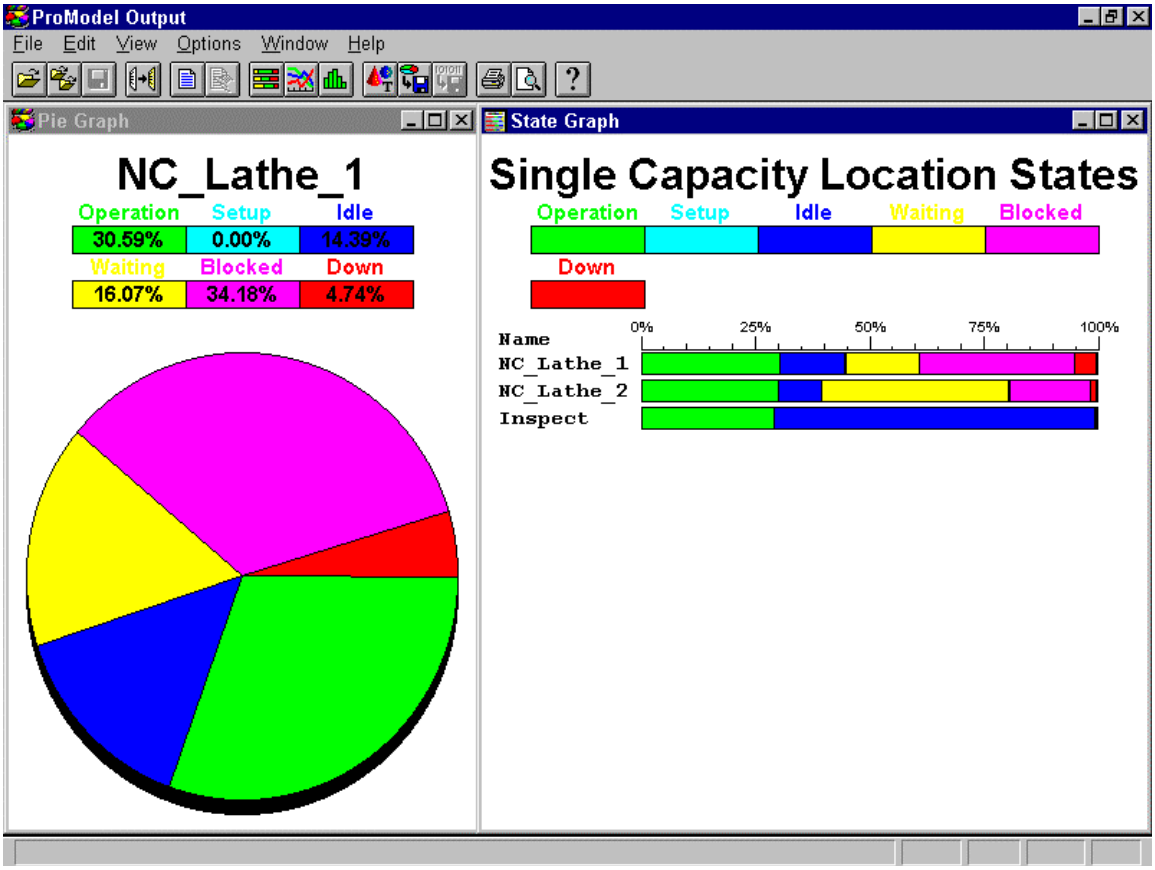

Figure 1: Graphical Representation of output in ProModel

ProModel will not be the chosen tool as it is a expensive tool to purchase and it requires the user to first be familiar with the program before it can be used. A model then has to be constructed in ProModel.

2.3.2. ARENA

Arena is a simulation tool which can be used to model a variety of systems including business and manufacturing processes (Arena Simulation Software Products, 2010). Different Arena editions are available depending on the intended use.

The following is an example of an Arena model:

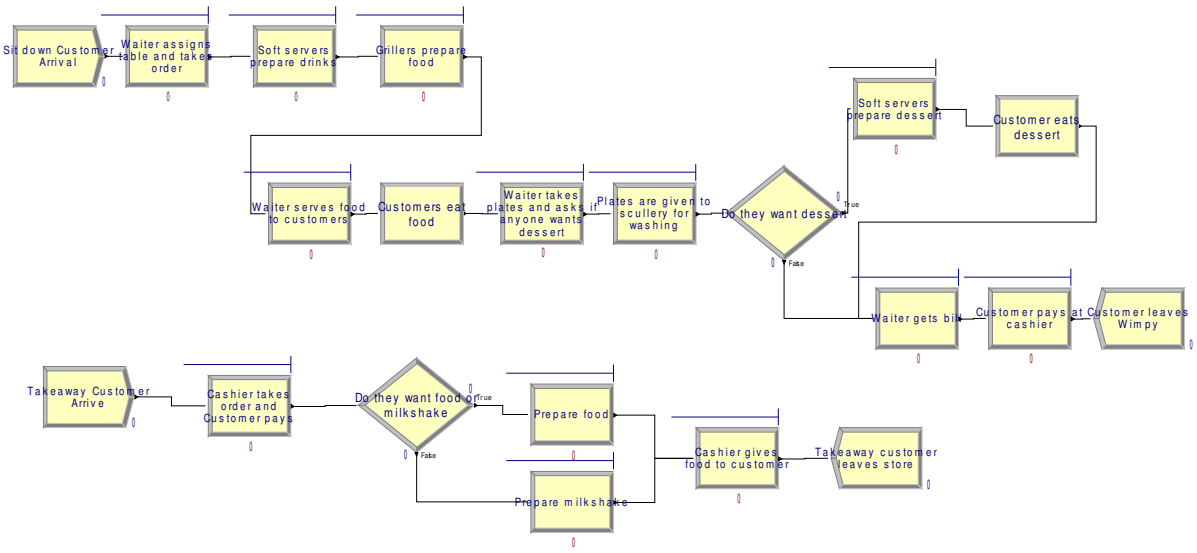

Figure 2: An example of an Arena model

2.3.3. MATLAB

Matlab is a computing language which has a wide variety of applications, including linear programming, data analyses, simulation and financial models. Matlab enables the user to solve problems faster than other programming languages, such as C++. Matlab also offers 2-D and 3-D graphical representation of data. (Matlab Product Description, 2010)

Matlab also have to be purchased and learned before it can be effectively applied to the problem. Therefore this tool will also not be the selected method.

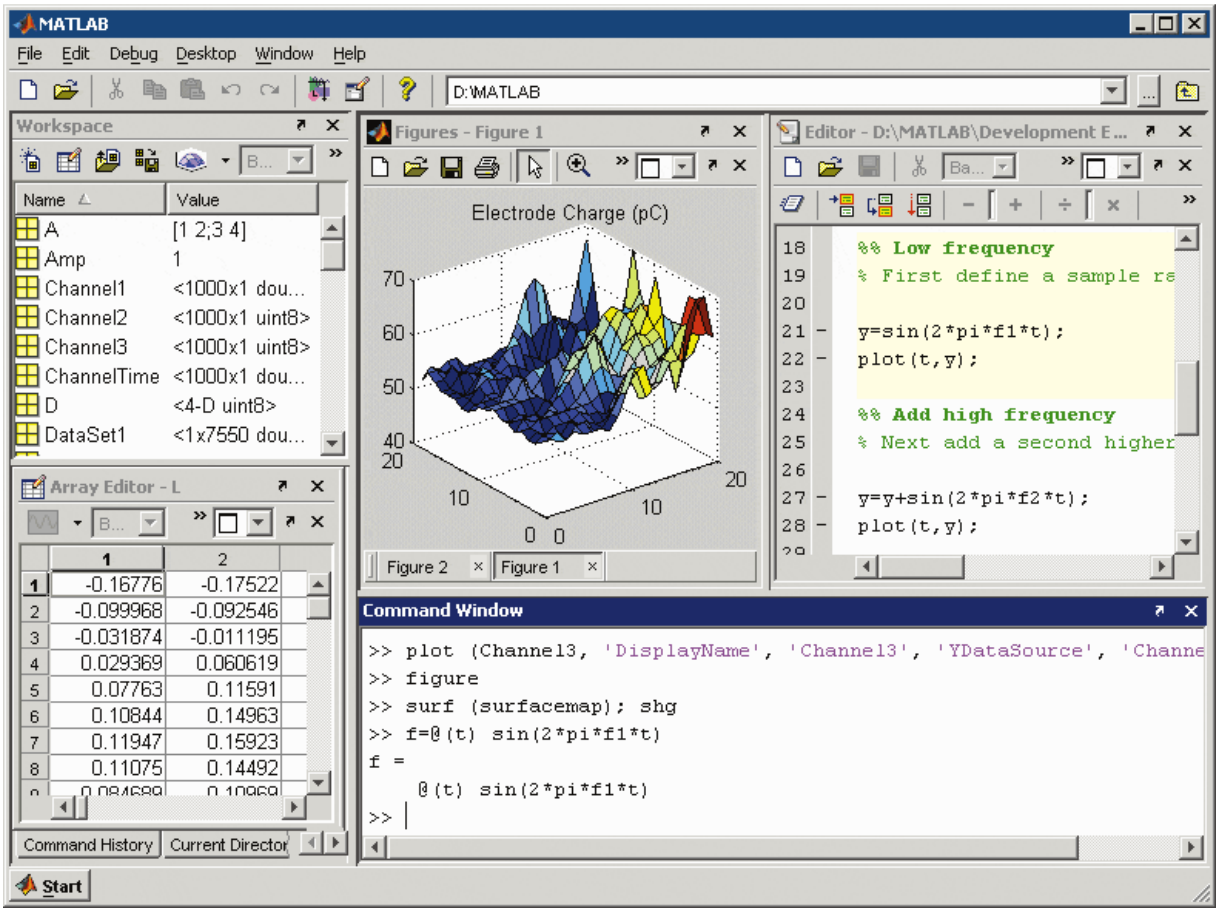

Figure 3: Matlab desktop

2.3.4. VISUAL BASIC APPLICATION IN EXCEL

Visual Basic Application (VBA) is a programming language for Microsoft Office. VBA in Excel enables the user to write commands to control aspects of the program. Excel VBA can be used for a variety of applications from analyzing data to executing complex models. (Walkenbach, 2007)

Walkenbach (2007) specifies a few advantages and disadvantages when using Excel VBA. This includes the following:

Advantages:

- Excel is consistent and executes commands in exactly the same way
- Excel performs tasks faster than a person can do the same tasks manually
- Programs will be performed without error, when a good program is written
- Programs can be written in such a form, so that a non-technical person will be able to use it.

Disadvantages:

- The user first has to be trained to write programs in VBA
- VBA is not a stand-alone program so each user has to have a copy of Excel
- The VBA program that is written by the user may not be applicable in all circumstances. This can produce errors. It is up to the user to be aware of this fact.
- Some code written in a version of Excel i.e. Excel 2007, may not be compatible with other versions. This is due to the fact that Excel is continuously upgraded.

VBA is the chosen tool because it is cost effective and adaptive. It is easy to use a VBA model as most computers have Microsoft Office Excel and therefore it is not necessary to purchase other expensive software and pay yearly subscriptions.

The following diagram illustrates the VBA program window:

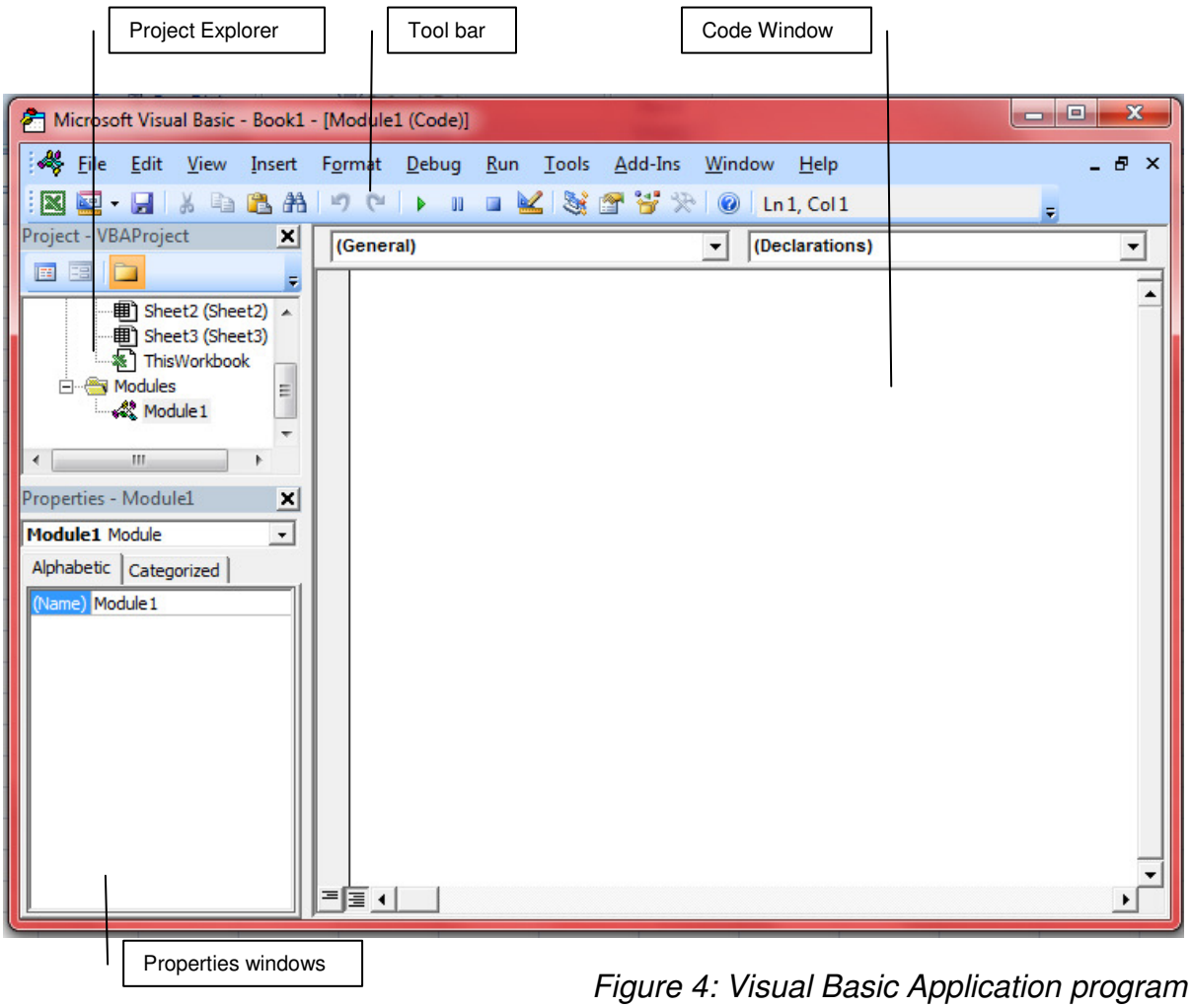

Figure 4: Visual Basic Application program

3. MODEL FORMATION

3.1 INTRODUCTION

As explained in the previous section, it was decided that Visual Basic Application (VBA) in Microsoft Office Excel will be used. First a few key terms and concepts will be explained, followed by a description of the model.

3.2 KEY TERMS AND CONCEPTS

Some key terms and concepts are explained below, which will then be used throughout this chapter.

Bin: Storage space in the warehouse

Bin Capacity: The amount of a certain part that the storage space can hold

Bin References: List of all the bins in the warehouse with the current inventory

Bin Types: Storage spaces that are different sizes

Empty Bin Policies: The method that is used to assign a part to an empty storage space in a section

Inventory: The parts that are currently in the warehouse

Overcapacity Policies: The method that is used when a section in the warehouse is full and parts have to be assigned to that section

Part Inflow: Part arrival at the warehouse. The parts then have to be assigned to a specific storage space

Part Number: A unique number that identifies every type of part

Part Outflow: Part exiting the system. Parts are taken out of the storage space

Part Quantity: The amount of a part that arrives at the warehouse

Part References: The amount of a part that can be added to a certain bin type. The SUT (please refer to the definition below) is referenced against the bin type to obtain the capacity.

Sections: The warehouse is divided into different areas. Each part is assigned to an area in the warehouse

Shortage Policies: The method used when a part is needed, but is not available in the warehouse

SUT: Storage Unit Type – A box that the parts arrive in. Each SUT has different dimensions.

Unallocated Parts: Parts that could not be placed in a storage space

3.3 USER FORM

The user form enables the user to add data and has a step by step wizard for first time users. The following diagram shows the main menu for the user form.

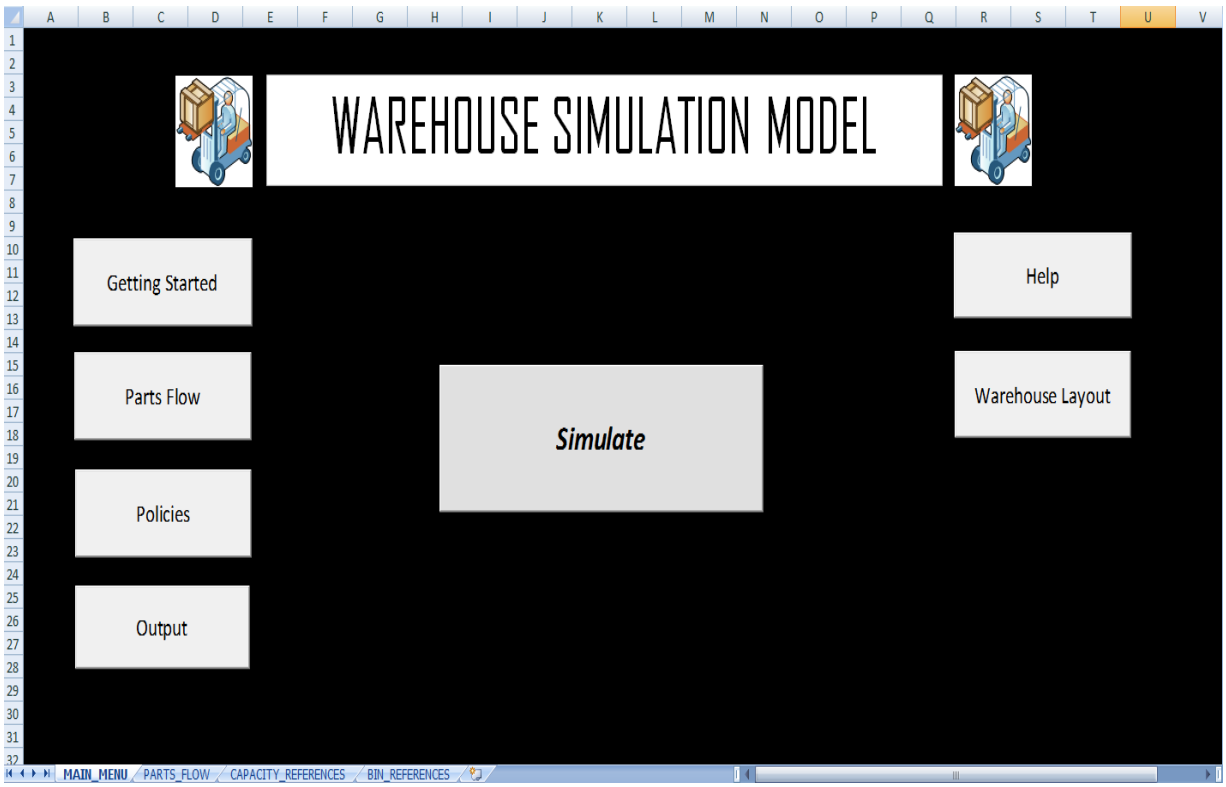

Figure 5: Main Menu

3.3.1. GETTING STARTED

The "Getting Started" is designed for first time users. It is a step by step process which enables the user to enter data in a user-friendly way. Some of the preliminary steps are explained below.

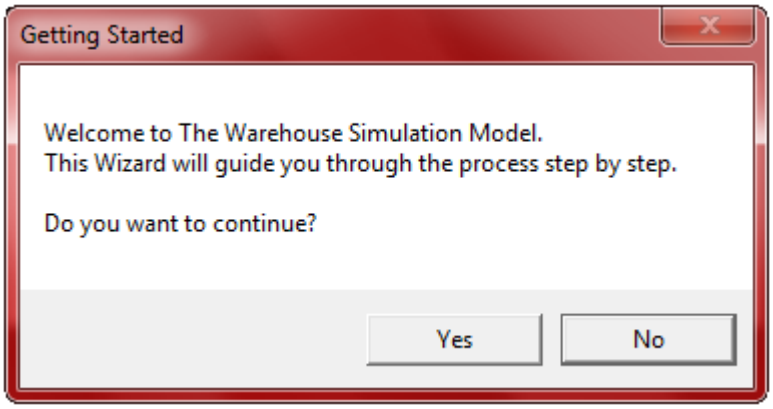

Figure 6: "Getting Started" Welcome Screen

If the user selects "yes":

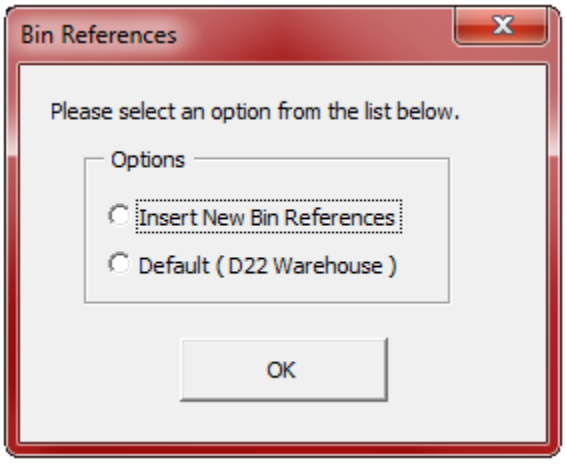

Figure 7: Bin References Selection Screen

The user can now choose to add new bin references or use the default warehouse. The default warehouse is Nissan's D22 warehouse, which is the subject of this project. The new bin references option is given to enable the user to simulate another warehouse or in the case where the bin references have changed in the current warehouse. The bin references are a list of the current bins in the warehouse with the current inventory.

Insert New Bin References:

	A	B	C	D	E	F	G	H	I
1	Getting Started Wizard			Main Menu					
2	SECTION	BINTYPE	BIN NAME	CONTAINING PART	SUT	COUNT			
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									

Figure 8: Bin References Worksheet

When the user chooses to enter new bin references, the model will be directed to the Bin References Worksheet. This will enable the user to add new bin references. When the user if finished, the “Getting Started” button is pressed. This will ensure that the user is returned to the wizard. If the user chooses to continue without the wizard, the “Finish” button is pressed. The model will then navigate to the Main Menu.

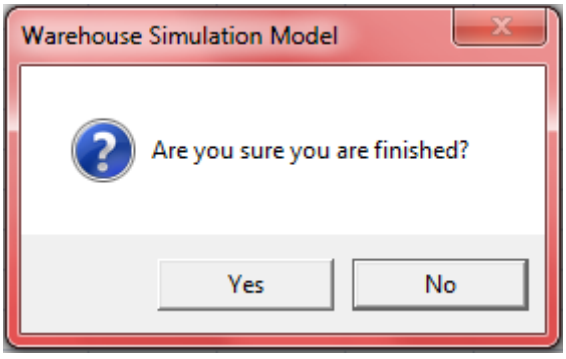

Figure 9: Message Box – “Are you sure…?”

This message box ensures that no problems occur if the “Getting Started” or “Finish” buttons are pressed by accident. If “yes” is selected the user will be directed back to the “main menu”, where the wizard will continue (if “Getting Started” was selected”).

Default:

The bin references will be added. The user will continue to the next step.

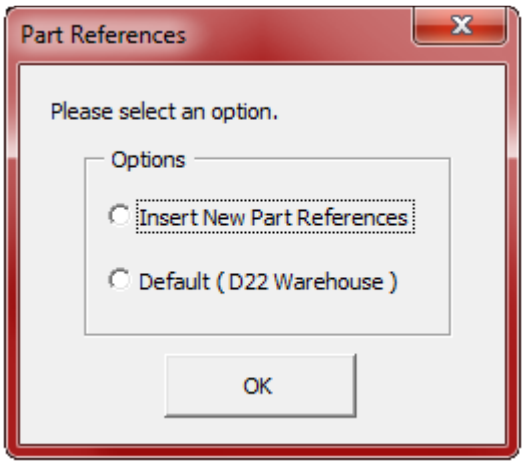

Figure 10: Parts References Dialogue Box

Insert New Part References:

The user is directed to the Capacity References worksheet. The part references can now be added.

	A	B	C	D	E	F	G
1	Getting Started Wizard	Main Menu					
2	SECTION	SUB SECTION	SUT	H1	H2	F1	F2
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							

Figure 11: Capacity References Worksheet

The user can now decide whether to continue with the Wizard or return to the Main Menu.

Default:

The default part references are left in place and the user continues to the next step.

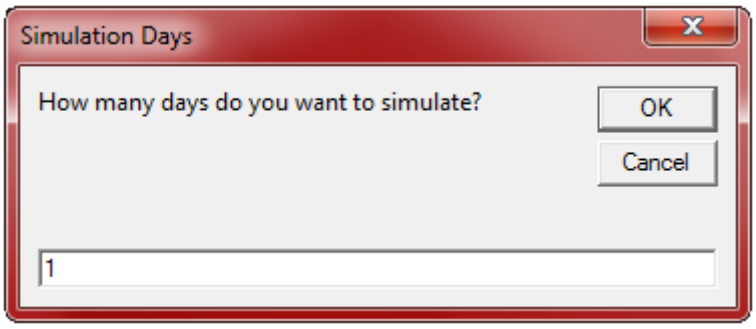

Figure 12: Simulation Days Dialogue Box

The user is asked for the amount of days he wishes to simulate. The default amount is one day. The days simulated is recorded.

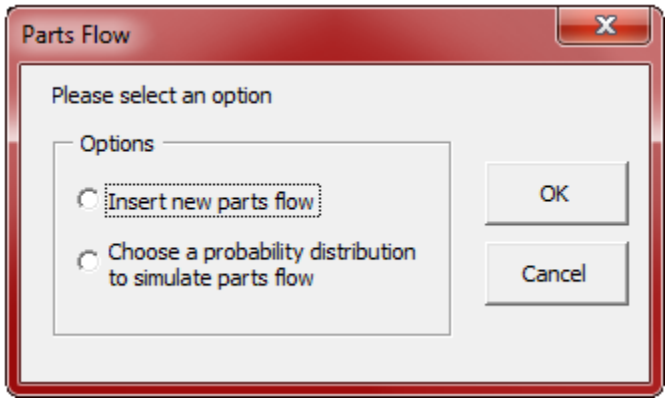

Figure 13: Parts Flow Dialogue Box

The user must also choose between inserting new parts flow and using a probability distribution to simulate this. The user will be asked to enter either the parts flow or choose the characteristics of the probability distribution.

New Parts Flow:

	A	B	C	D	E	F	G	H	I
1	Getting Started Wizard		Main Menu	Day1		Day2		Day3	
2	PARTNUMBER	SECTION	SUT	Inflow	Outflow	Inflow	Outflow	Inflow	Outflow
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									

Figure 14: Parts Flow Worksheet

The days are automatically entered to ensure the user enters the right data. “Three days” was entered to illustrate this.

At this point the model will add all the Section worksheets to the model. This data is sourced from the Capacity References worksheet.

After the user is finished, he returns to the Wizard.

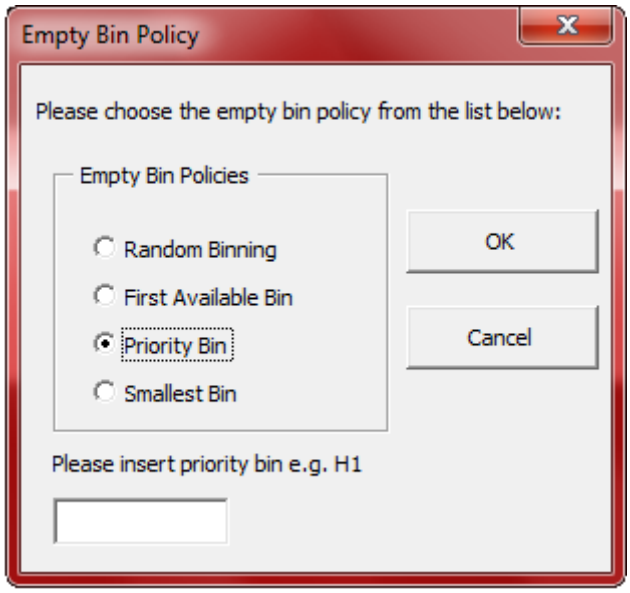

Figure 15: Empty Bin Policy Dialogue Box

The user must select the preferred empty bin policy. The empty bin policy is then recorded depending on the selection the user makes. The policies are recorded as:

- “Random”
- “First”
- “Priority”
- “Smallest”

When the user selects “Priority Bin”, the insert box will appear to ask the user for the priority bin.

The user the moves on to the next step:

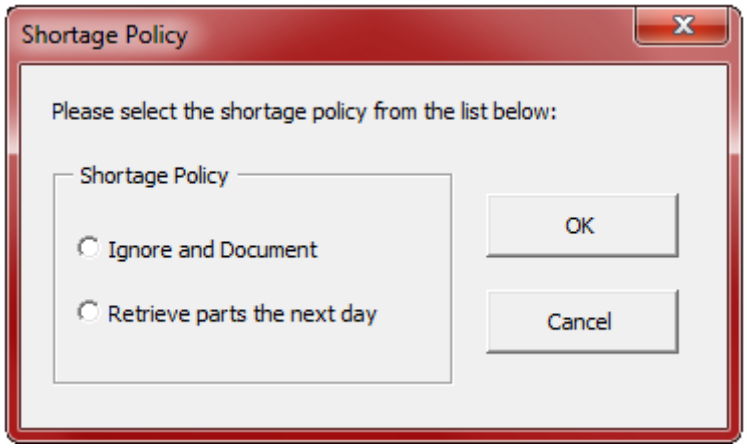

Figure 16: Shortage Policy Dialogue Box

The shortage policy is chosen and is recorded as:

- “Ignore”
- “NextDay”

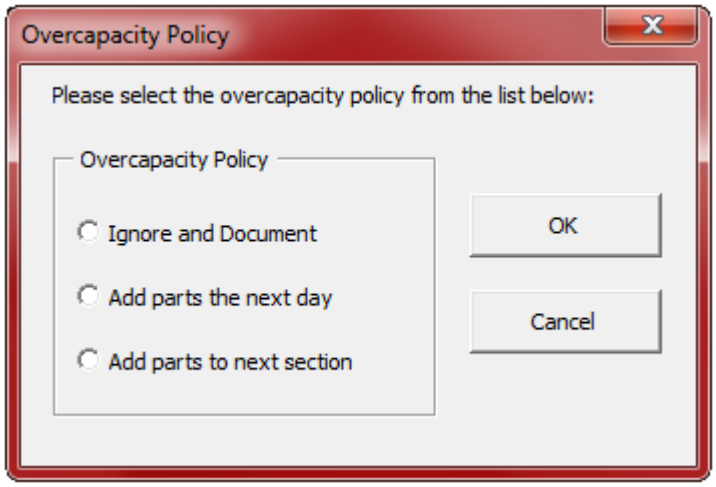

Figure 17: Overcapacity Policy Dialogue Box

The overcapacity policy is recorded as one of the following:

- “Ignore”
- “NextDay”
- “NextSection”

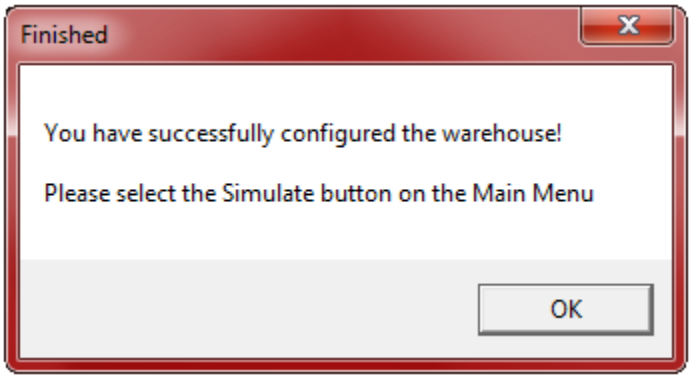

Figure 18: Finished Dialogue Box

The Wizard is now finished. The user is requested to press the “Simulate” button on the Main Menu as shown below:

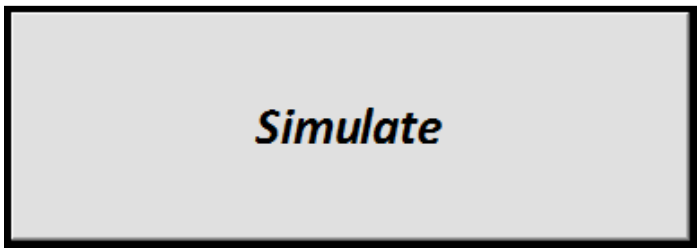

Figure 19: Simulate Button

3.3.2. PARTS FLOW

The “Parts Flow” button on the Main Menu (Fig. 5) allows a user to make adjustments only to the parts flow, without having to repeat all the steps in the “Getting Started” process.

3.3.3. POLICIES

All the policies (shortage, overcapacity and empty bin) can be changed by using this button. (Fig. 5) The following dialogue box will appear:

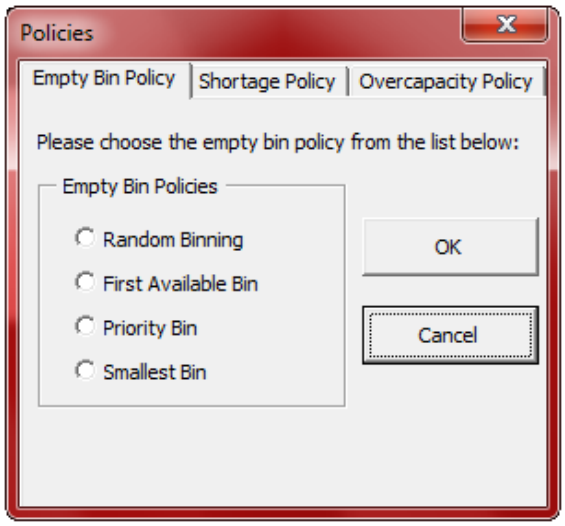

Figure 20a: Policies – Empty Bin Policy Tab

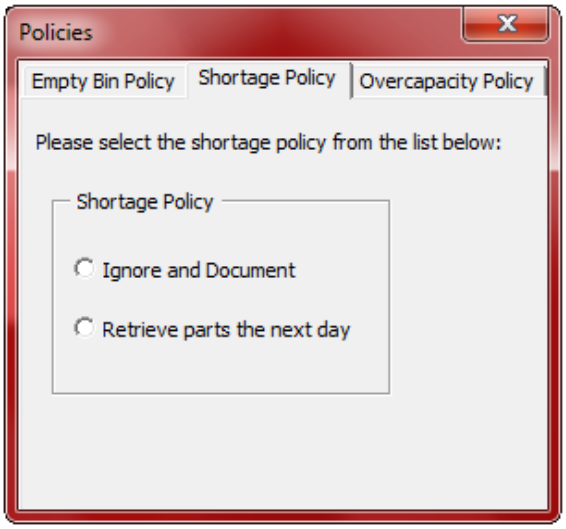

Figure 20b: Policies – Shortage Policy Tab

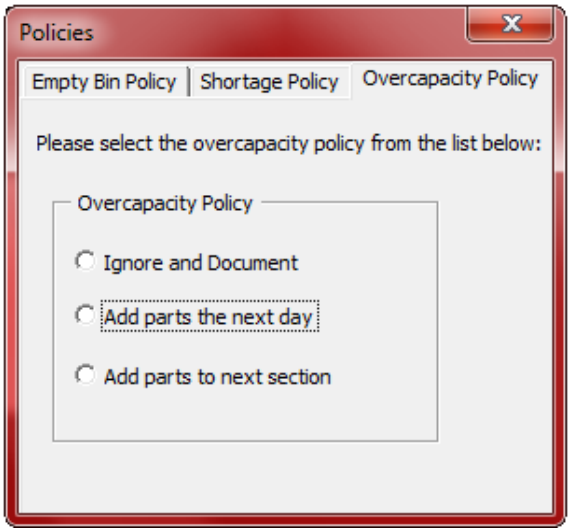

Figure 20c: Policies – Overcapacity Policy Tab

3.3.4. OUTPUT

The outputs can be viewed by pressing this button. (Fig. 5)

3.3.5. HELP

The “Help” button shows the user definitions and general tips regarding the model.

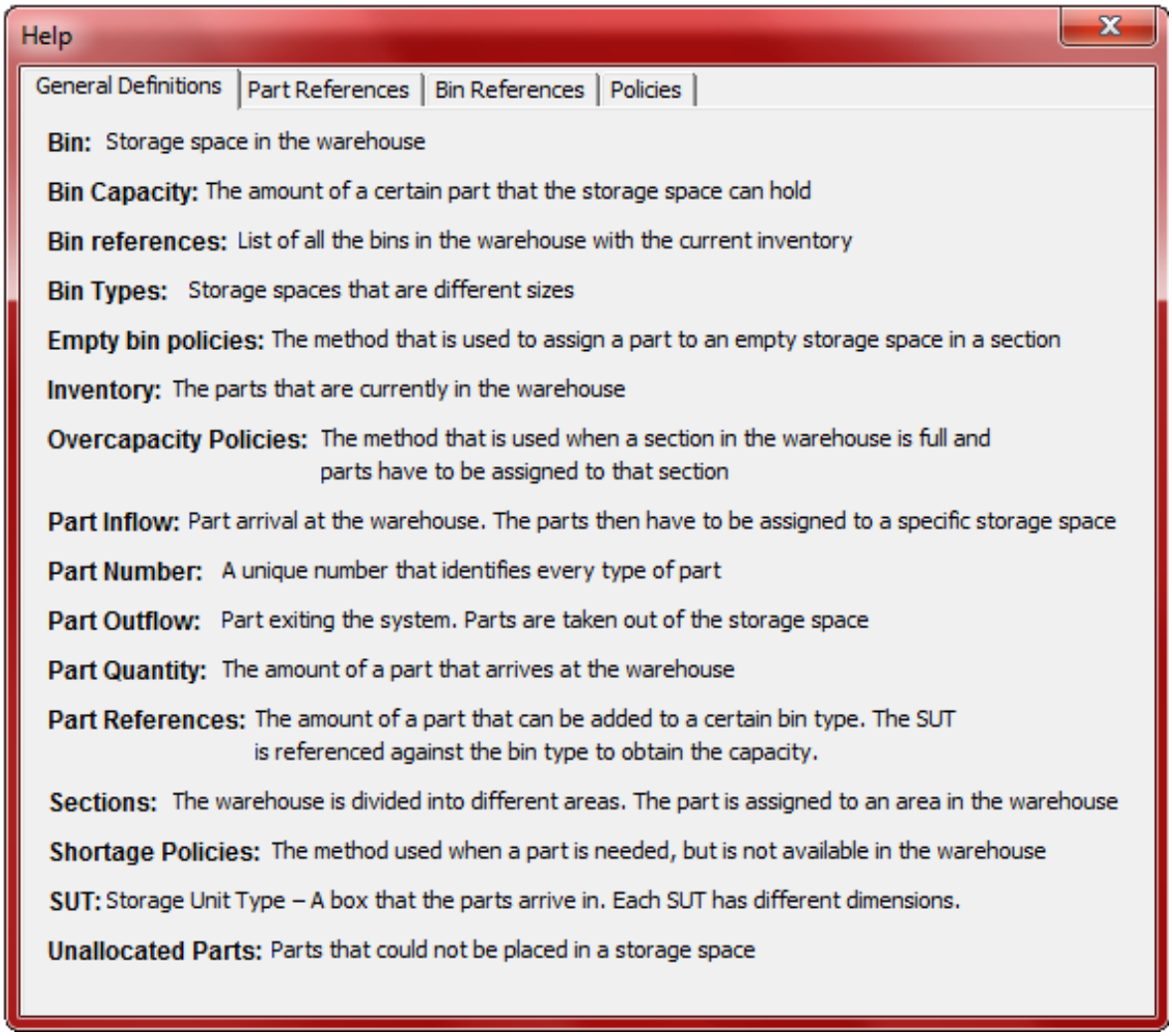

Figure 21: Help File

3.3.6. WAREHOUSE LAYOUT

The user can enter the warehouse layout and configuration in this process.

3.3.7. SIMULATE

When the user has inserted all the variables needed, he can now simulate the data. All the variables will be displayed to make sure the user is aware of all the selections he has made.

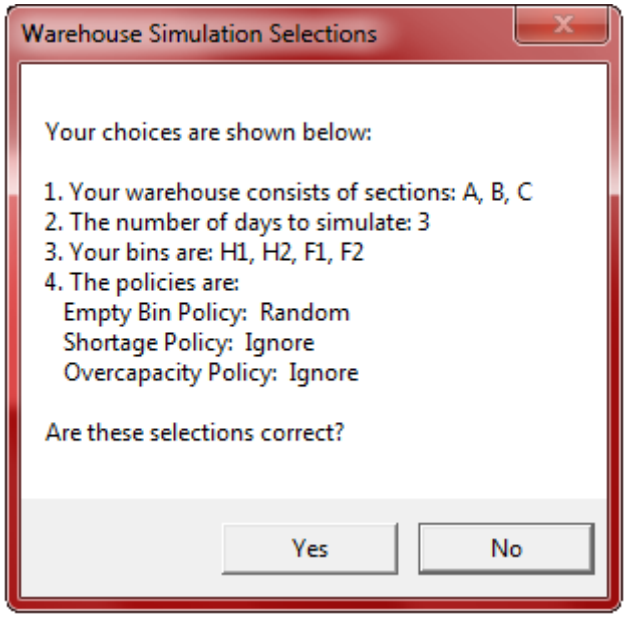

Figure 22: User's Choices

3.4 PARTS INFLOW

The following diagram illustrates the parts flow including the empty bin policies:

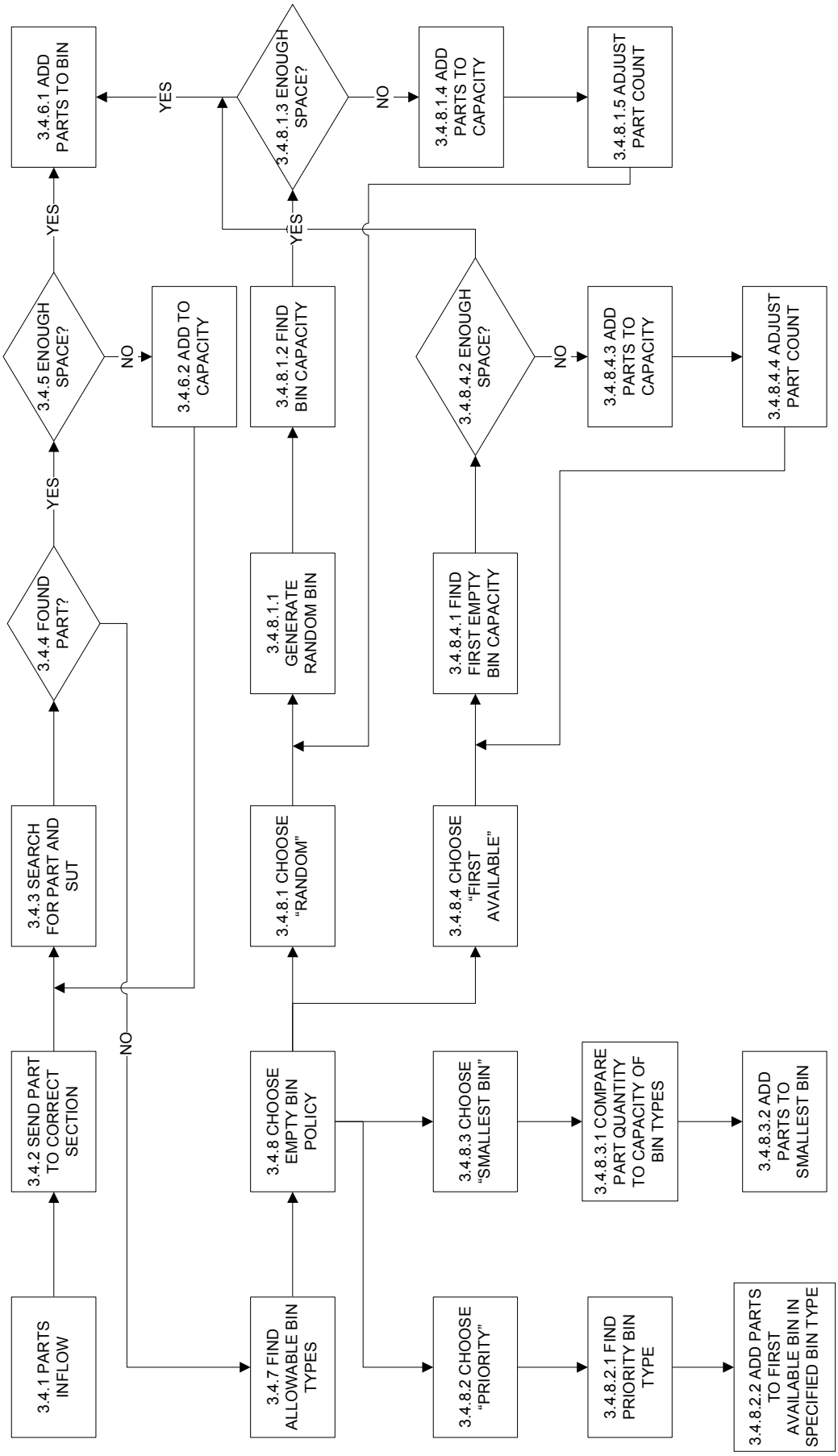

Figure 23: Parts Inflow Diagram

Refer to Appendix A for full Visual Basic Application code.

3.4.1. PARTS FLOW

A certain amount of a part arrives at the warehouse for storage. A part has a part number, SUT, section and allowable bin types.

3.4.2. SEND PART TO CORRECT SECTION

The parts are then sent to the correct section depending on the type of part it is.

3.4.3. SEARCH FOR PART AND SUT

The model then searches for that specific part in the section by cross referencing the part number and SUT to parts that are currently present. It is important for the SUT to match as well, because some parts can be packaged in different size boxes (SUT's).

3.4.4. PART FOUND?

Has the part been found in the section? Yes/ No

3.4.5. ENOUGH SPACE?

If the part has been found in the section, the model has to test if there is enough space in the bin.

3.4.6. ADD PARTS TO BIN

Yes: If there is enough space in the bin, the parts are added.

$$Bin\ Count = Bin\ Count\ Starting + Part\ Quantity$$

$$Part\ Quantity = 0$$

No: If there is not enough space available the parts are added until the bin capacity is reached.

$$Part\ Quantity = Part\ Quantity\ Starting - Parts\ added$$

$$Bin\ Count = Bin\ Capacity$$

3.4.7. FIND ALLOWABLE BIN TYPES

When part containing bins are full or parts could not be found in the section, parts have to be added to empty bins. Parts can only be added to certain bin types, thus the model has to determine the allowable bin types for the specific part.

3.4.8. CHOOSE EMPTY BIN POLICY

There are different methods to assign parts to empty bins. The methods are random, priority, smallest bin and first available. The methods are explained below.

3.4.8.1. CHOOSE "RANDOM"

The random empty bin policy places a part in the section in an allowable random bin.

- *Generate random bin:* a random bin number is generated
- *Find bin capacity:* the chosen bin's capacity is determined by cross referencing the bin type and the part's SUT.
- *Enough space?* Yes/No
- *Add parts to capacity:*

  Not enough space – add parts until bin capacity is reached.

  Enough space – refer to 3.4.6

- *Adjust part quantity:* refer to 3.4.6 "No"

The method is then repeated until the part quantity is equal to zero.

3.4.8.2. CHOOSE "PRIORITY"

Priority empty bin policy allows the user to give preference to one specific bin type e.g. give preference to hand bins or forklift bins.

- *Find priority bin type:* the model determines the bin type that has the highest priority and then second highest and so forth. This will determine the order in which the model will search for empty bins.
- *Add parts to first available bin:* An empty bin is located that corresponds to the selected bin type. Parts are then added as in 3.4.6.

3.4.8.3. CHOOSE "SMALLEST BIN"

The smallest bin empty bin policy aims to make the part allocation more optimal. The model will search for the smallest bin in the section that will be able to take all the parts of a specific part flow instance.

- *Compare part quantity to capacity:* The bin types are referenced to the part's SUT to determine the capacity. These capacities are then compared to the part quantity. The bin type with the smallest capacity with adequate space to hold all parts is then selected.
- *Add parts to smallest bin:* Parts are added in the same manner as 3.4.6.

3.4.8.4. CHOOSE "FIRST AVAILABLE"

This empty bin policy finds the first available bin in the section.

- *Find first available bin capacity:* The first empty bin in the section is found.
- *Enough space:* Yes/No
- *Add parts to capacity:* Refer to 3.4.6 "No"
- *Adjust part count:* Refer to 3.4.6 "No"

The overcapacity policy is shown below.

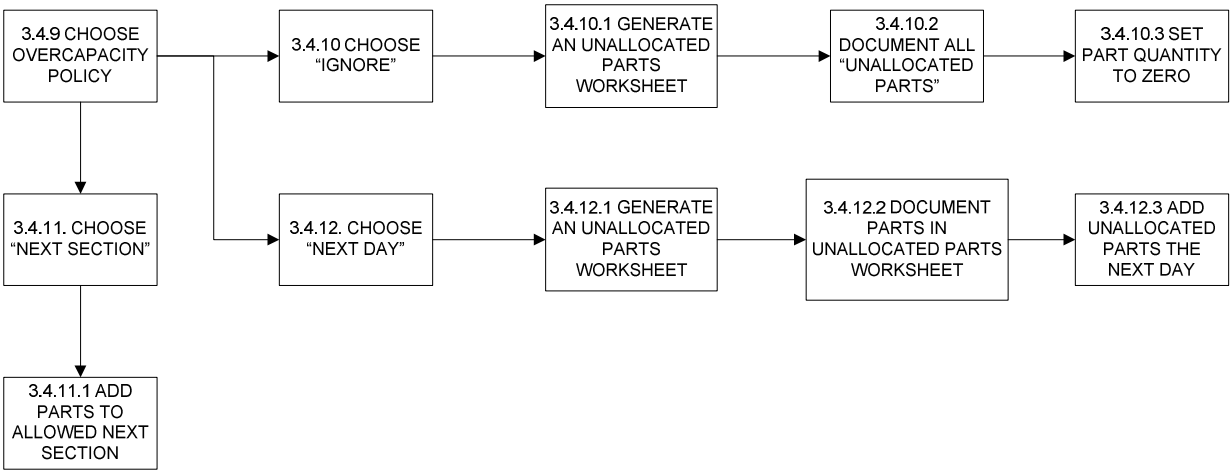

Figure 24: Overcapacity Policy Process Flow

3.4.9. CHOOSE OVERCAPACITY POLICY

An overcapacity policy takes effect when a section in the warehouse has reached capacity. The policies inform the model what to do with parts that cannot be

assigned to a bin. There are three policies to choose from namely; ignore, next section and next day.

3.4.10. CHOOSE OVERCAPACITY POLICY

The ignore policy only documents the part as “unallocated parts”.

3.4.10.1. GENERATE AN “UNALLOCATED PARTS” WORKSHEET

A worksheet is generated, which will document these parts.

3.4.10.2. DOCUMENT ALL “UNALLOCATED PARTS”

All parts that could not be allocated to a bin is documented to be used in later investigations.

3.4.10.3. SET PART QUANTITY TO ZERO

The part quantity is then set to zero and in effect ignored in terms of bin allocation.

3.4.11. CHOOSE “NEXT SECTION”

This policy allocates parts to the next section when that current section is full.

3.4.11.1. ADD PARTS TO ALLOWED NEXT SECTION

The parts are added to the next section in the same manner as described above in 3.4.1 to 3.4.8.

3.4.12. CHOOSE “NEXT DAY”

In this policy the unallocated parts are documented. The unallocated parts are then allocated the next day. This refers to the model’s time periods. These parts are first added before the day’s parts flow begins. This simulates a situation where parts are just stacked somewhere in the warehouse until some parts have left the system.

3.4.12.1. GENERATE AN “UNALLOCATED PARTS” WORKSHEET

An unallocated parts worksheet is generated to document the parts.

3.4.12.2. DOCUMENT PARTS IN “UNALLOCATED PARTS”

Parts are documented in the worksheet.

3.4.12.3. ADD UNALLOCATED PARTS THE NEXT DAY

The parts are added the “next day” and the records are deleted.

3.5 PARTS OUTFLOW

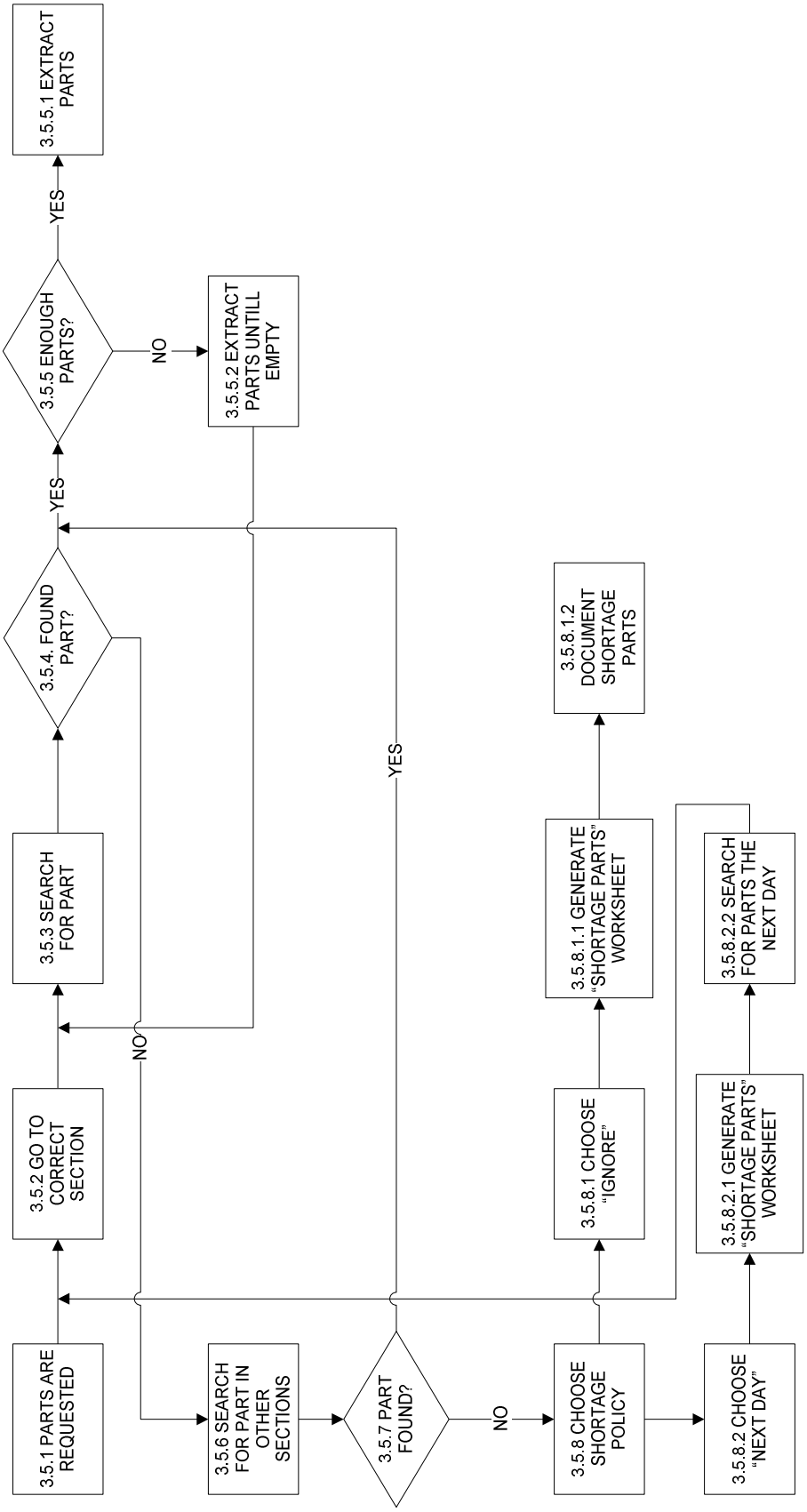

Figure 25: Parts Outflow Diagram

Search for the part by cross referencing the part number and SUT to the parts already in the section.

3.5.1. PARTS ARE REQUESTED

The production line requests parts from the warehouse.

3.5.2. GO TO CORRECT SECTION

Search for the part in the section that it has been allocated to previously.

3.5.3. SEARCH FOR PART

3.5.4. FOUND PART?

Yes/No

3.5.5. ENOUGH PARTS?

If the part has been found the model must now check if there are enough parts available.

3.5.5.1. EXTRACT PARTS

Requested parts are extracted if there are enough parts available.

$$Bin\ Count = Bin\ Count\ Starting - Part\ Quantity\ Requested$$

$$Part\ Quantity = 0$$

3.5.5.2. EXTRACT PARTS UNTILL EMPTY

If there are not enough parts available the bin is emptied.

$$Bin\ Count = 0$$

$$Part\ Quantity = Part\ Quantity\ Starting - Parts\ Extracted$$

3.5.6. SEARCH FOR PART IN OTHER SECTIONS

If the part is not found in the section and the “next section” overcapacity policy was chosen, the model will search for the part in other sections.

3.5.7. FOUND PART?

If the part was found refer to procedure 3.5.5.

3.5.8. CHOOSE SHORTAGE POLICY

The shortage policies take effect when there are no requested parts in the warehouse. There are two policies namely “ignore” and “next day”.

3.5.8.1. CHOOSE “IGNORE”

The ignore policy just documents the shortage parts.

- *Generate “shortage parts” worksheet*: A shortage parts worksheet is generated to document shortages that can be used for later analysis.
- *Document shortage parts:* Record shortage parts.

3.5.8.2. CHOOSE “NEXT DAY”

The parts are located the next day when more parts have already been allocated to the warehouse.

- *Generate “shortage parts” worksheet*
- *Search for parts the next day:* Follow same procedure as in 3.5.5.

3.6 WAREHOUSE LAYOUT

The user will be asked to insert the warehouse layout and configuration. Microsoft Office Visio will be used. The layout will be linked to data in the excel model so that both programs are up to date.

The top view of the warehouse is entered. When the user double-clicks on the specific rack, he is redirected to the side view of that rack.

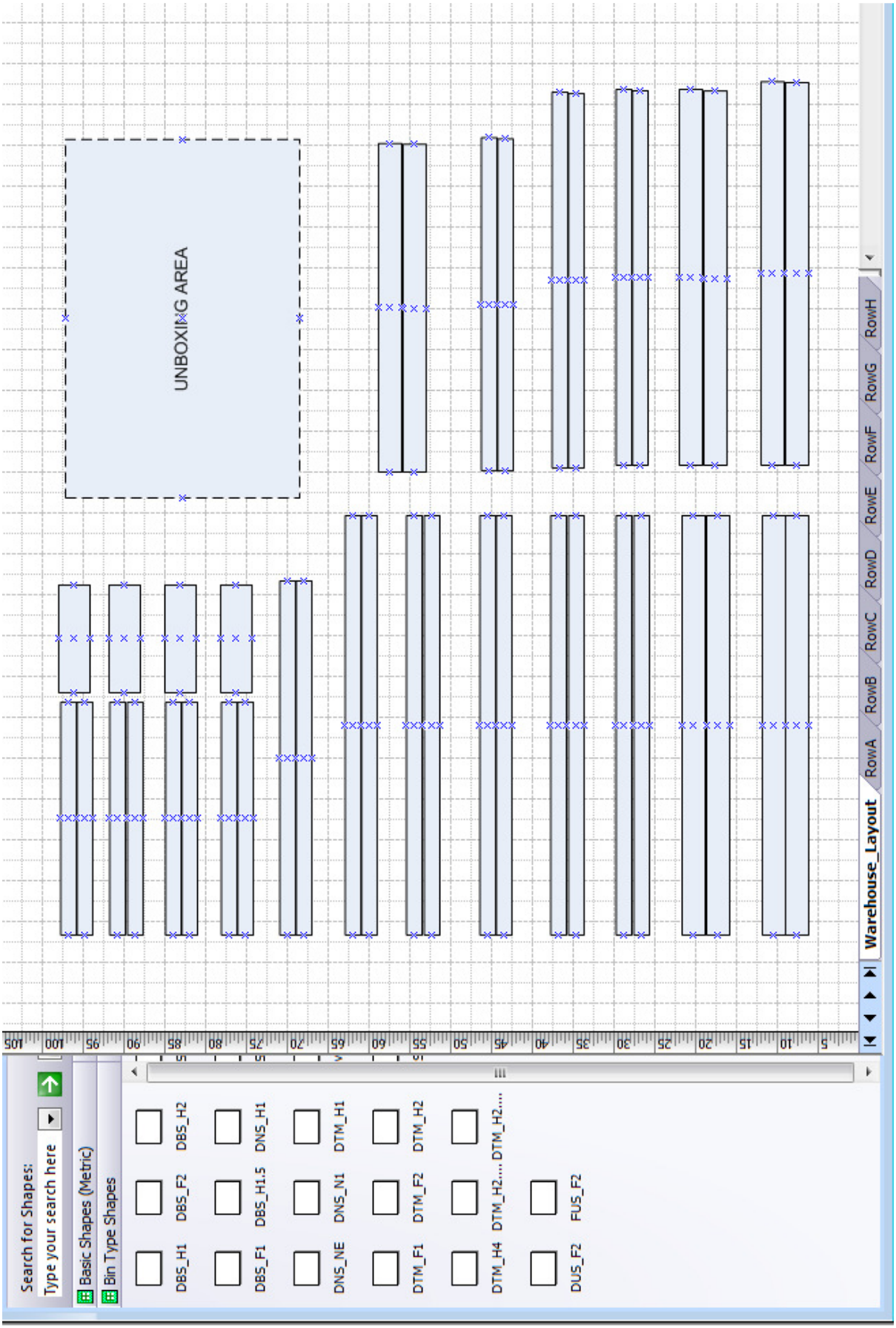

Figure 26: Visio Warehouse Layout

Below is an example of custom bins that the user can utilize in order to enter the layout.

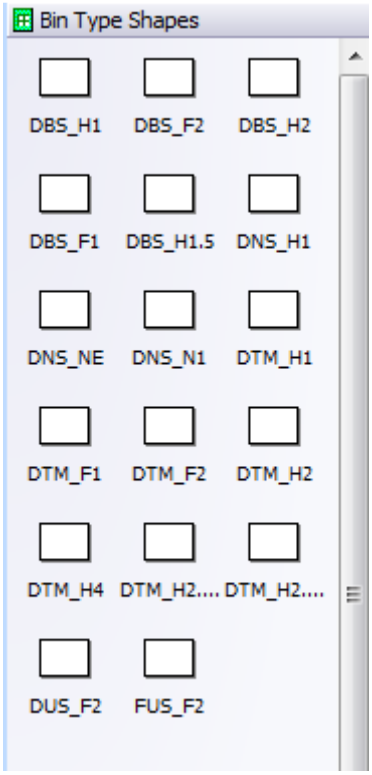

Figure 27: Bin Type Shapes

The shapes have data graphics that show the user how full the bin is. The degrees are shown below:

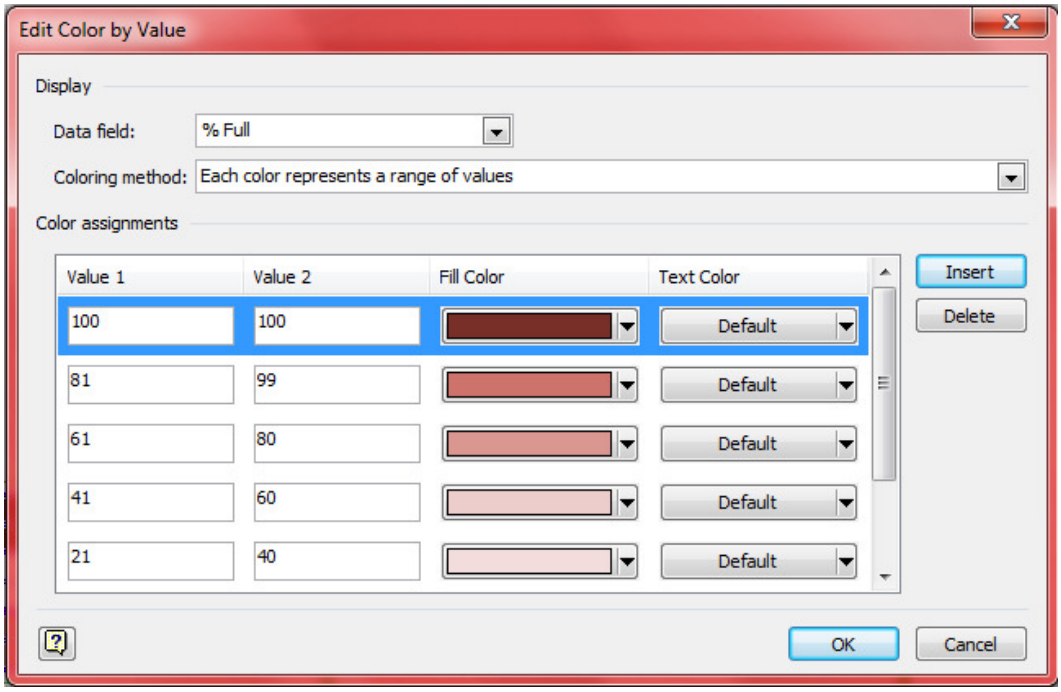

Figure 28: Data Graphic Ranges

Each shape has characteristics that are unique to that shape. The characteristics can include section, bin type and dimensions.

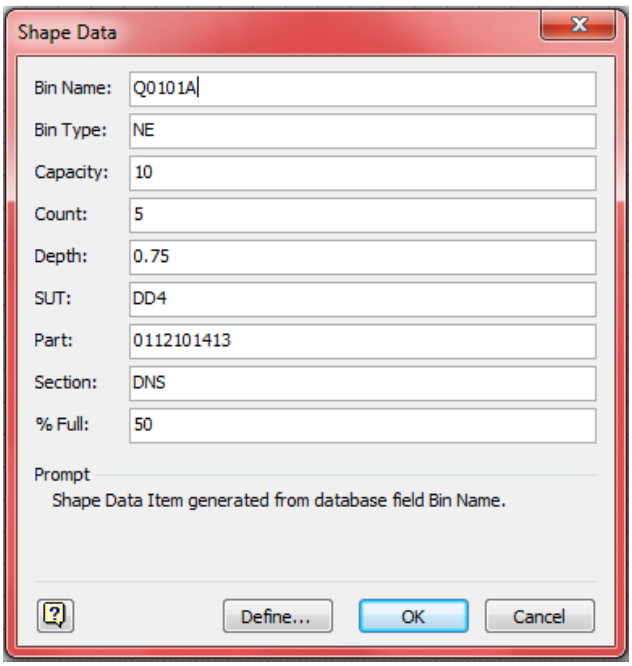

Figure 29: Bin Shape Data

3.7 OUTPUTS

Some outputs of the model will be:

- Available empty bins in each bin type, section and the warehouse
- Full bins in each bin type, section and the warehouse
- Partially full bins
- Fluctuations in capacities

3.8 CONCLUSION

The model formation was explained in Chapter 3. We looked at the user forms, as well as the parts inflow and outflow. A few simulations were run and the computation results will be explained in Chapter 4, in terms of assumptions, objectives, inputs and outputs.

4. COMPUTATION RESULTS

4.1 ASSUMPTIONS

General Assumptions

- The warehouse already exists
- All dimensions are known
- All data is available (bins, sections, parts flow etc.)

4.2 OBJECTIVES

The objective of this model is to simulate the warehouse inflows, outflow and bin allocations. After results have been computed the user can now use these results to conduct a “what if” analysis. The model is used a decision tool to determine the layout that suites the user requirements.

4.3 INPUTS

The simulation model requires some input values before it can be run.

The inputs are:

4.3.1. WAREHOUSE LAYOUT (OPTIONAL)

Please refer to *Figure 26* for the D22 warehouse layout.

Below is a side view of Row Q in Section DNS:

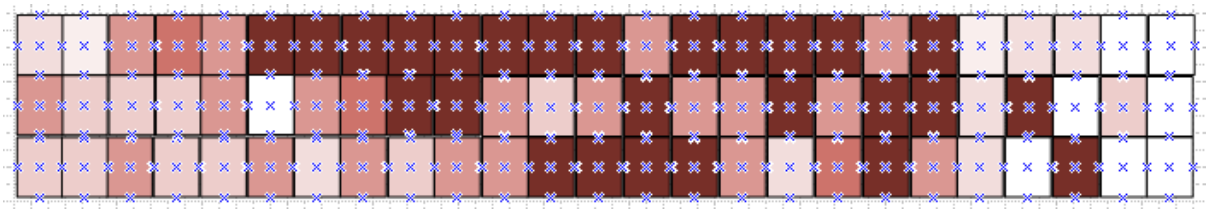

Figure 30: Row Q Bins

From the figure above, the user can clearly see the empty, partially full and full bins.

Each bin has unique characteristics. An example is shown below:

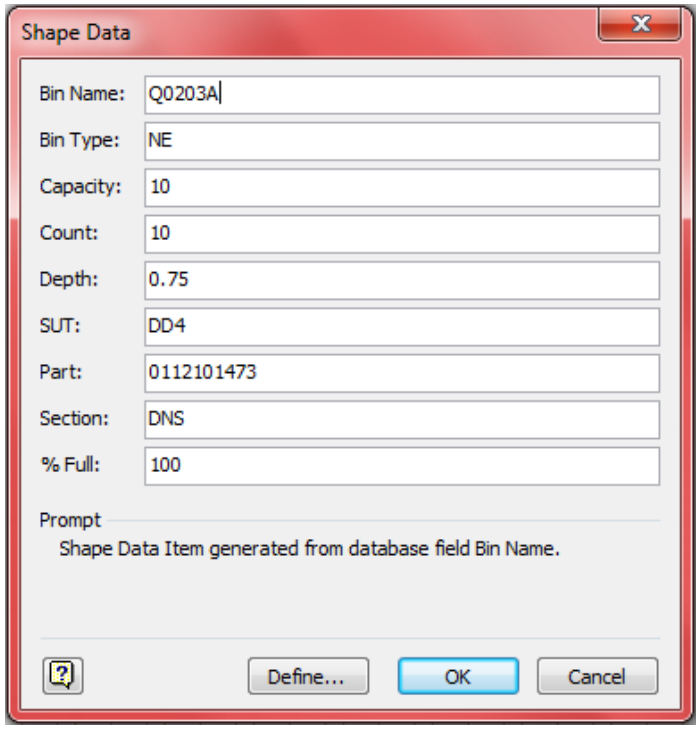

Figure 31: Bin Q0203A Shape Data

4.3.2. WAREHOUSE CONFIGURATION

- *Sections*
  The warehouse consists of sections:
  1. DNS
  2. DTM
  3. DUS
  4. DBS
  5. DLO
  6. DBK
- *Bin types*
  The bin types are:
  1. H1
  2. H2
  3. H3
  4. H4
  5. N1
  6. NE
  7. F1
  8. F2
  9. F3
  10. BK

- *Parts*
  A list of parts with inflows and outflows was entered into the model.
- *Bin capacities*
  Bin references were added to the model.

4.3.3. BIN REFERENCES (STARTING INVENTORY)

A part of the bin references are shown below:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1								BINTYPE					
2	SECTION		SUT	H1	H2	H3	H4	N1	NE	BK	F1	F2	F3
3	DBK		C03							5			
4	DBK		C04							10			
5	DBK		C05							10			
6	DBK		C06							8			
7	DBK		C07							10			
8	DBK		C08							8			
9	DBK		C09							8			
10	DBK		R07							10			
11	DBK		R14							10			
12	DBK		R17							10			
13	DBK		RA							10			
14	DBK		RAA							10			
15	DBK		RAK							10			

Figure 32: Bin References

4.3.4. INFLOW AND OUTFLOW OF PARTS

A part of the inflow and outflow is shown below:

	A	B	C	D	E	F	G	H	I	J
1	PART	SECTION	SUT	DAY 1		DAY 2		DAY 3		
2	NUMBER			IN	OUT	IN	OUT	IN	OUT	
3	2709079964	DTM	DD4	1	3	2	2	1	5	
4	87611VJ20A	DTM	DD9	17	3	13	14	19	10	
5	92480VK505	DTM	RC1	2	3	4	2	2	1	
6	74960VK90A	DTM	RC1	6	5	1	1	7	1	
7	170401S40A	DUS	DD9	1	1	3	2	3	3	
8	0112101413	DNS	DD4	1	1	3	1	5	5	
9	089129441A	DNS	DD4	4	1	1	3	8	3	
10	46289VK010	DNS	RC4	1	3	3	2	3	2	
11	803312S402	DTM	DD9	1	2	1	1	2	4	
12	081270201E	DNS	DD4	1	2	3	2	1	1	
13	0112101473	DNS	DD4	9	5	5	7	10	4	
14	0112100022	DNS	DD4	3	2	2	3	4	2	
15	081210351E	DNS	DD4	1	3	2	2	1	5	
16	081210251E	DNS	DD4	1	1	1	1	5	4	
17	081268201G	DNS	DD4	1	1	3	2	3	1	
18	40262S0400	DNS	DD4	1	2	3	2	5	1	
19	54215VL00B	DNS	DD4	1	3	1	2	4	4	
20	743982S400	DNS	DD4	14	6	13	2	14	8	

Figure 33: Parts Flow

4.4 OUTPUTS

A few simulations were run in order to show how the model works.

Inputs that remained constant:

- Parts flow
- Bin references
- Simulation days: 3
- Part references

Variable inputs:

- Empty Bin Policy

4.4.1. SCENARIO 1: "RANDOM" EMPTY BIN POLICY

Some of the outputs are shown below:

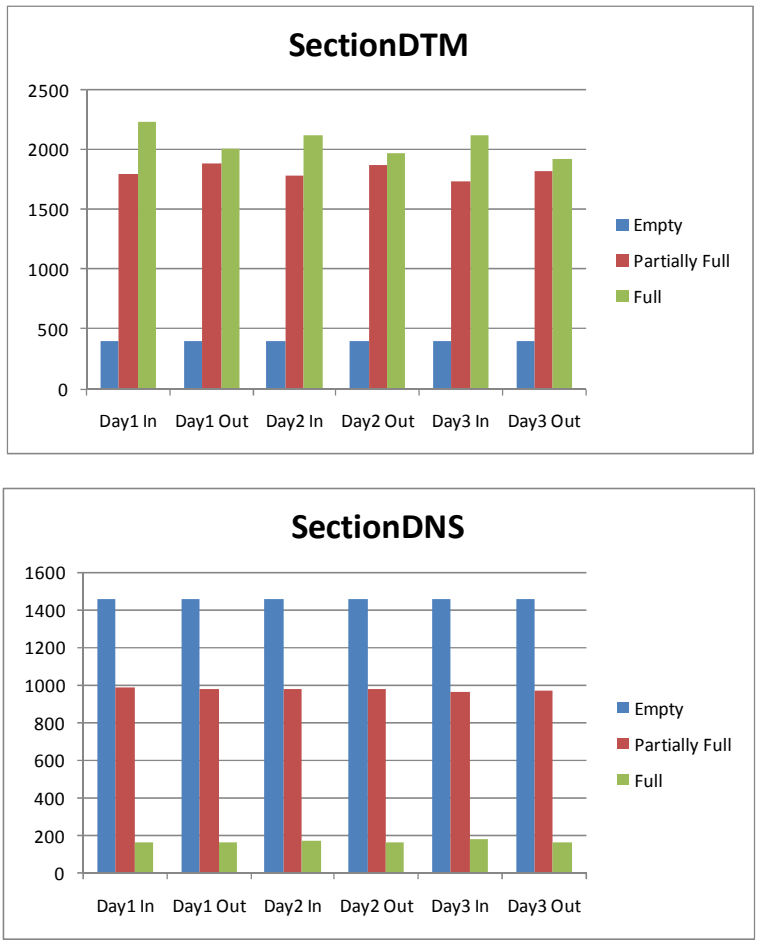

Figure 34: "Random": Section Capacity Fluctuation

It can be seen from comparing the two graphs that Section DTM is very full compared to Section DNS. From this the option of expanding the DTM section into Section DNS can be investigated.

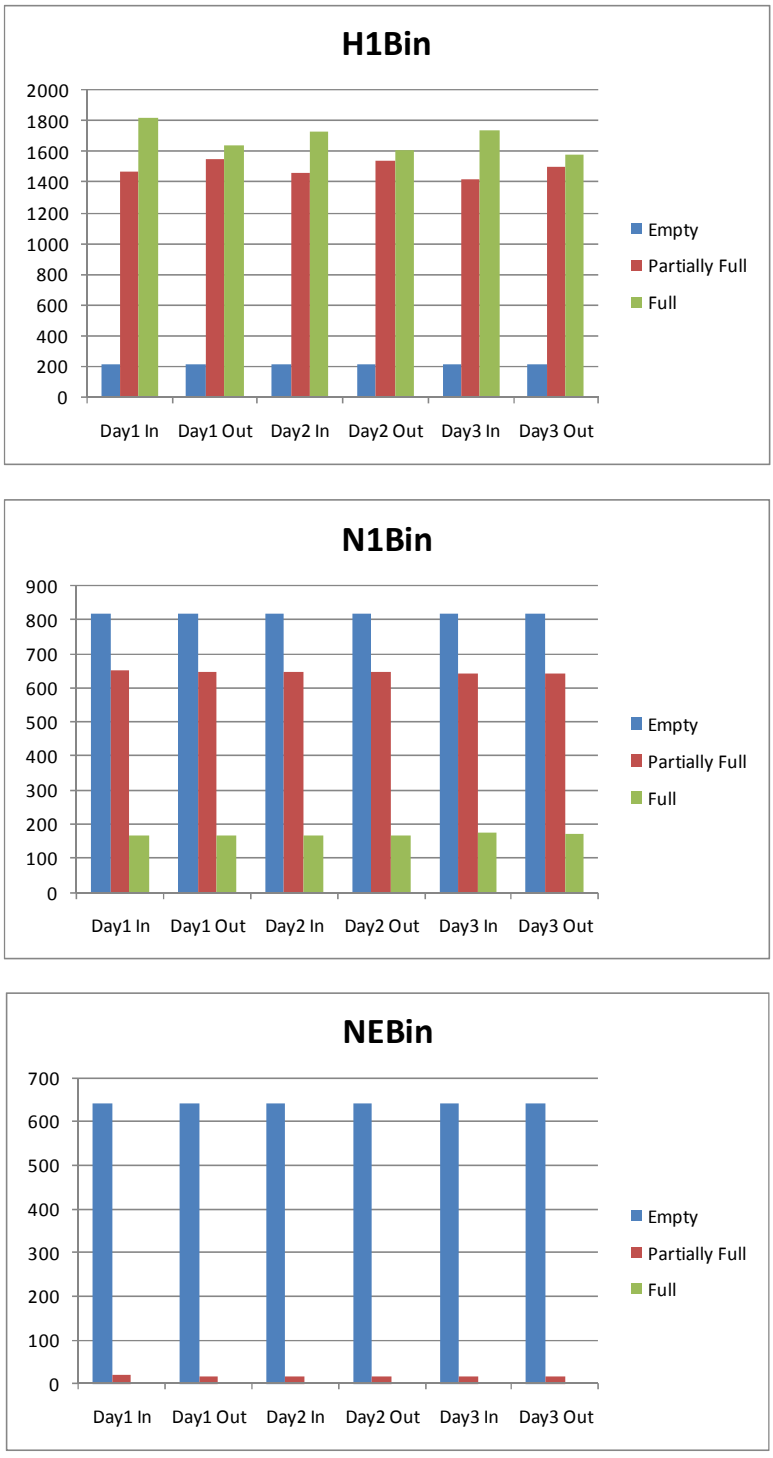

Figure 35: "Random": Bin Type Capacity Fluctuation

From the graphs it is seen that H1 is over-used and that the other two bin types have a large number of empty bins. A proposed solution is to convert the N1 and NE bins to H1 bins.

4.4.2. SCENARIO 2: "FIRST AVAILABLE" EMPTY BIN POLICY

The DBK section are shown below:

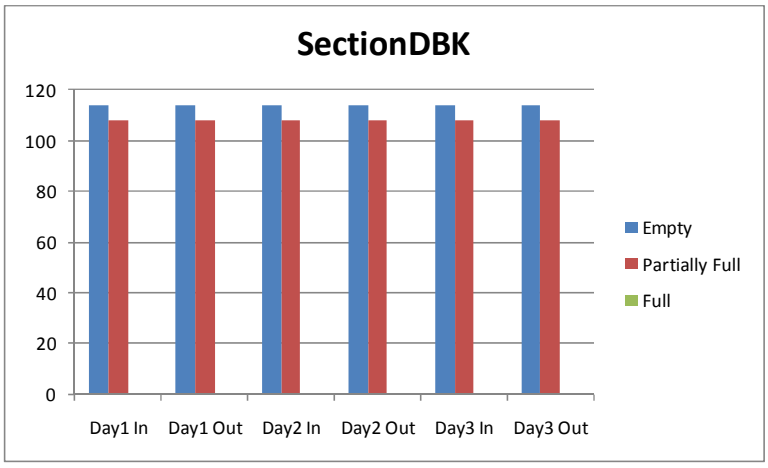

Figure 36: "First": Section Capacity Fluctuation

If this section continues to have more than half the space empty, that space must be reassigned to other sections.

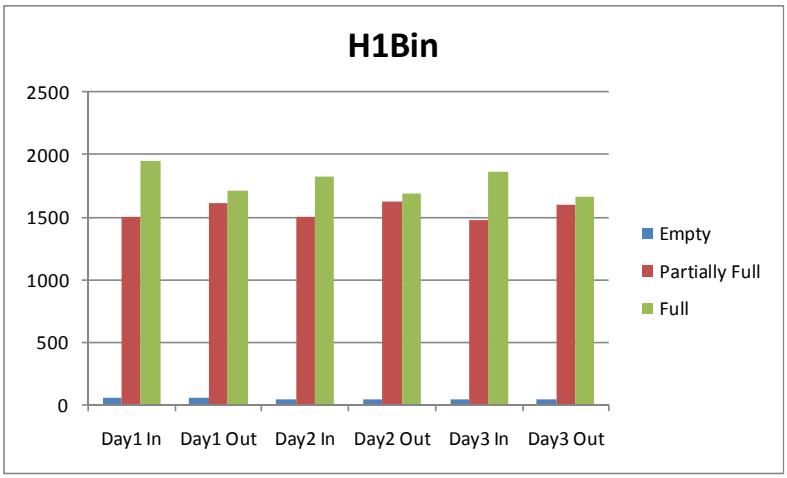

Figure 37: "First": Bin Type Capacity Fluctuation

The H1 bin type continues to be mostly full, even more so than when "Random" binning is used. It is clear that H1 bins must be increased.

4.4.3. SCENARIO 3: "PRIORITY" EMPTY BIN POLICY

In order to decrease the usage of H1 bins it was decided that the priority bin "F1" is used.

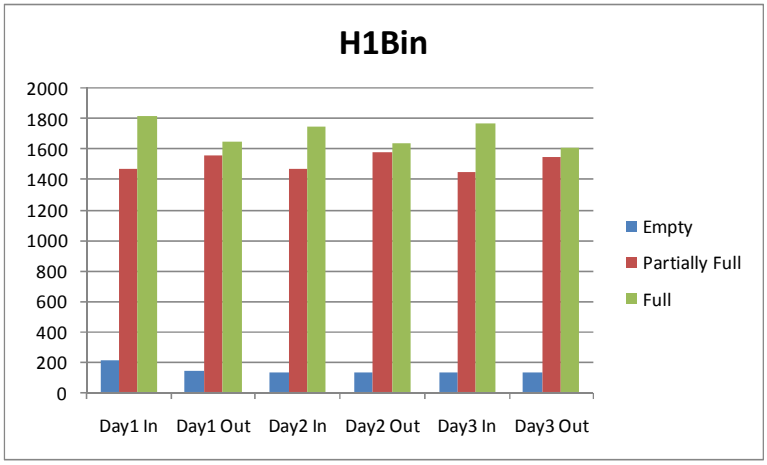

Figure 38: "Priority": Bin Type Capacity Fluctuation

It improved slightly compared to the "First" binning policy, because there are parts that cannot be allocated to F1 bins. This is further evidence that H1 bins need to be increased.

The user can change the different policies or warehouse configuration to analyse the differences.

4.5 CONCLUSION

In Chapter 4 some scenarios were modelled and compared. It was shown that the model can be used to compare different scenarios in order to make the appropriate changes. In Chapter 5 some future work is discussed.

5. FUTURE WORK AND CONCLUSION

After the completion of the project some extension to the model can be evaluated. Some extensions include the following:

5.1 POLICIES

A wider variety of empty bin, shortage and overcapacity policies can be added.

5.2 OTHER WAREHOUSES

Further modification of the model can include modelling other types of warehouses, such as:

- Warehouses with different inventory i.e. liquids
- Warehouses with fixed binning (Each part can only go to a specific bin)
- Warehouses with partial fixed binning (Each part can only go to a few specific bin)

5.3 USERFORM

The user form can be improved to make it more user-friendly. The outputs that are available can also be extended to include a more comprehensive analysis of the partially full bins. The partially full bins can be measured in detail i.e. amount of bins between 20% and 40%.

5.4 OTHER APPLICATIONS

The possibility of other applications can be investigated. Other applications can include:

- Warehouses in other sectors
- Shops (i.e. grocery/ clothing)

REFERENCES

[1] Arena simulation software products, viewed 3 May 2010, <http://www.arenasimulation.com/Products_Products.aspx>

[2] Baker, P and Canessa, M. Warehouse Design: A structured approach. *European Journal of Operational Research 193 (2009) 425-436,* <http://www.sciencedirect.com>

[3] Deterministic model definition, viewed 28 April 2010, <http://www.businessdictionary.com/definition/deterministic-model.html>

[4] Kelton, W.D., Sadowski, R.P., Sturrock,D.T. *Simulation with Arena, Fourth Edition,* McGraw-Hill International Edition, 2007.

[5] Macro, J.G. and Reino, E.S. A simulation tool to determine warehouse efficiencies and storage allocations, *Proceedings of the 2002 Winter Simulation Conference, E. Yücesan, C.-H. Chen, J. L. Snowdon and J. M. Charnes, Eds.*

[6] Matlab product description, viewed 3 May 2010, <http://www.mathworks.com>

[7] Nissan Corporate Information, viewed 15 March 2010, <http://www.nissan.co.za/en/web/header/header_1351.htm>

[8] ProModel products, viewed 30 April 2010, http://www.promodel.com

[9] Rouwenhorst , B , Reuter, B, Stockrahm , V, van Houtum , G.J., Mantel, R.J., Zijm, W.H.M. Warehouse design and control: Framework and literature review, *European Journal of Operational Research 122 (2000) 515-533*.

[10] The Actuarial Education Company (2010). *Actuarial Risk Management, Chapter 29: Modelling.*

[11] Walkenbach, J. *Excel 2007 VBA Programming for Dummies.* Wiley Publishing, Inc.

APPENDIX A

1. PARTS FLOW

```
Public PartNumber As String
Public SUT As String
Public Section As String
Public PartQuantity As Integer
Public Day As String
Public DataColumn As Integer

Sub FlowOfParts()

Dim PartRow As Integer
Dim FlowColumn As Integer
Dim RowLocal As Integer 'This is used within a section
Dim ColumnLocal As Integer
Dim RefRow As Integer
Dim RefColumn As Integer
Dim CheckColumn As Integer
Dim DayCount As Integer
Dim ShortagePolicy As String
Dim EmptyBinPolicy As String
Dim OvercapacityPolicy As String

'FlowColumn is the column starting from the amount of part inflow/outflow
FlowColumn = 4
DayCount = 1
DataColumn = 2

'Go to Parts Flow worksheet
Worksheets("PARTS_FLOW").Select

'Go through all days partflow
Do Until IsEmpty(Cells(3, FlowColumn))

PartRow = 3

    'Go through one days part flow
    Do Until IsEmpty(Cells(PartRow, 1))
```

```
'Get values
PartNumber = Worksheets("PARTS_FLOW").Cells(PartRow, 1)
Section = Worksheets("PARTS_FLOW").Cells(PartRow, 2)
SUT = Worksheets("PARTS_FLOW").Cells(PartRow, 3)
PartQuantity = Worksheets("PARTS_FLOW").Cells(PartRow, FlowColumn)

Select Case FlowColumn Mod 2

    'This represents parts inflow
    Case Is = 0

    'Get Day Value
    Day = "Day" & DayCount & "In"

    'If the overcapacity policy is next day
    If OvercapacityPolicy = "NextDay" Then

        Call OverCapacityNextDayFlow

    End If

    'Go to the correct Section and define variables
    Worksheets("SECTION_" & Section).Select
    RowLocal = 3

        'Find part in the section
        Do Until IsEmpty(Cells(RowLocal, 1)) Or PartQuantity = 0

              If Cells(RowLocal, 2) = PartNumber And Cells(RowLocal, 3)
              = SUT Then

            Count = Cells(RowLocal, 4)
            Capacity = Cells(RowLocal, 5)
            AvailableSpace = Capacity - Count

                'Check if there is space in bin
                Select Case AvailableSpace

                    'Not enough space available in bin
                    Case Is < PartQuantity
                        PartQuantity = PartQuantity - AvailableSpace
```

```
                Cells(RowLocal, 10) = AvailableSpace
                Cells(RowLocal, 4) = Capacity
                Cells(RowLocal, 6) = 100

            'Enough space available in bin
            Case Is >= PartQuantity
                Cells(RowLocal, 4) = Count + PartQuantity
                Cells(RowLocal, 10) = PartQuantity
                  Cells(RowLocal, 6) = (Count + PartQuantity) /
                  Capacity * 100
                PartQuantity = 0


        End Select

    End If

RowLocal = RowLocal + 1

Loop


'If there are still parts left
If PartQuantity > 0 Then

    Select Case EmptyBinPolicy

        Case Is = "Random"
            'Place parts randomly
            Call EmptyBinPolicy1

        Case Is = "First"
            'Bin Type Priority
            Call EmptyBinPolicy2

        Case Is = "Priority"
            'Smallest bin
            Call EmptyBinPolicy3

        Case Is = "Smallest"
            'First available bin
```

```
                    Call EmptyBinPolicy4

            End Select

        End If

    'This represents parts outflow
    Case Is = 1

    'Get Day Value
    Day = "Day" & DayCount & "Out"

    'If the shortage policy is next day
    If ShortagePolicy = "NextDay" Then
        Call ShortageNextDayFlow
    End If

'If next section policy is chosen extract parts in other sections
first
    If OvercapacityPolicy = "NextSection" Then
        Call OverCapacityNextSectionOutflow
    End If

    'Go to the correct Section
    Worksheets("SECTION_" & Section).Select
    RowLocal = 3


        'The program now searches for the part within the section
        Do Until IsEmpty(Cells(RowLocal, 1)) Or PartQuantity = 0

        If Cells(RowLocal, 2) = PartNumber And Cells(RowLocal, 3) = SUT
        Then

            Count = Cells(RowLocal, 4)
            Capacity = Cells(RowLocal, 5)

                Select Case PartQuantity

                    'Not enough parts in bin or just enough
                    Case Is >= Count
```

```
                    PartQuantity = PartQuantity - Count
                    Cells(RowLocal, 2) = "<<empty>>"
                    Cells(RowLocal, 3) = " "
                    Cells(RowLocal, 4) = " "
                    Cells(RowLocal, 5) = " "
                    Cells(RowLocal, 6) = " "

                'Enough parts in bin
                Case Is < Count
                    Cells(RowLocal, 4) = Count - PartQuantity
                    Cells(RowLocal, 6) = Cells(RowLocal, 4) /
                Cells(RowLocal, 5) * 100
                    PartQuantity = 0

             End Select

        End If

    RowLocal = RowLocal + 1
    Loop

    'When there are shortages
    If PartQuantity > 0 Then

        Select Case ShortagePolicy

            'Ignore and dokument
            Case Is = "Ignore"
                Call ShortagePolicy1

            'Carry over shortages to the next day
            Case Is = "NextDay"
                Call ShortagePolicy2

        End Select
    End If


End Select

PartRow = PartRow + 1
```

```
    'Go back to partsflow worksheet
    Worksheets("PARTS_FLOW").Select
    Loop

    'Call the output module for capacity fluctuations
    Call CapacityFluctuations


'Go back to partsflow worksheet
Worksheets("PARTS_FLOW").Select

'Go to the next set of parts flow
FlowColumn = FlowColumn + 1
DataColumn = DataColumn + 1

    'Go to next day
    If FlowColumn Mod 2 = 0 Then
        'Count to find the next day
        DayCount = DayCount + 1
    End If

Loop

'Create charts capacity fluctuations
Call CreateCharts


End Sub
```

2. EMPTY BIN POLICIES

"RANDOM"

```
Sub EmptyBinPolicy1() 'Place part randomly

Dim RefRow As Integer
Dim PartNumber As String
Dim Section As String
Dim SUT As String
Dim PartQuantity As Integer
Dim Number As Integer
Dim CapacityBinArray() As Integer
Dim BinArray() As String
Dim EmptyArray() As Integer
Dim R1 As Integer


'Select the reference worksheet
Worksheets("CAPACITY_REFERENCES").Select

RefRow = 3
i = 4

'Find the amount of bins
Do Until IsEmpty(Cells(2, i))

    i = i + 1
    BinTypeCount = BinTypeCount + 1

Loop

ColumnCount = 0
i = 4

Do Until IsEmpty(Cells(RefRow, 1)) Or PartQuantity = 0

    If Cells(RefRow, 1) = Section And Cells(RefRow, 3) = SUT Then
```

```
    'Count number of non empty cells in the rows thus amount of allowable
    bins
   AllowableCount =
    Application.WorksheetFunction.CountA(Range(Cells(RefRow, 4),
Cells(RefRow, BinTypeCount + 4)))

        'Assign array dimension
        ReDim CapacityBinArray(1 To AllowableCount)
        ReDim EmptyArray(1 To AllowableCount)
        ReDim BinArray(1 To AllowableCount)



        iCount = 1

        'Find the allowable bins
        Do Until ColumnCount = BinTypeCount

              If Worksheets("CAPACITY_REFERENCES").Cells(RefRow, i) > 0
              Then

                'Assign array values
                CapacityBinArray(iCount) = Cells(RefRow, i)
                BinArray(iCount) = Cells(2, i)

                'Get number of emptybins in that section and bintype
                Bin = Cells(2, i)
                'Match bin to give location row number
               Row1 = Application.WorksheetFunction.Match(Bin,
              Worksheets("SECTION_" & Section).Range("A:A"), 0)

              'Start counting from the next row because thats where
              bins start
                RowL1 = Row1 + 1
                'Select correct worksheet
                Worksheets("SECTION_" & Section).Select

                'Count the bins in that bintype
                Do Until IsEmpty(Cells(RowL1, 2))
```

```
            Count1 = Count1 + 1

            RowL1 = RowL1 + 1
        Loop

       EmptyArray(iCount) =
       Application.WorksheetFunction.CountIf(Range(Cells(Row1 +
       1, 2), Cells(Row1 + Count1, 2)), "<<empty>>")


        iCount = iCount + 1
    End If

ColumnCount = ColumnCount + 1
i = i + 1
Worksheets("CAPACITY_REFERENCES").Select
Loop

'Find the total empty bins
EmptyT = Application.WorksheetFunction.Sum(EmptyArray)

If EmptyT > 0 Then

    BinV = Application.WorksheetFunction.RandBetween(1, EmptyT)

    'Counter
    Counter = 1

    EmptySum = EmptyArray(1)
    Lowerbound = 1
    EmptyCount = 0
    Number = BinV

    Do Until Counter > AllowableCount Or PartQuantity = 0

        If BinV >= Lowerbound And BinV <= EmptySum Then

            Capacity = CapacityBinArray(Counter)
```

```
Bin = BinArray(Counter)
R1 = Application.WorksheetFunction.Match(Bin,
Worksheets("SECTION_" & Section).Range("A:A"), 0)
Worksheets("SECTION_" & Section).Select

    'Find the n'th empty bin
    Do Until EmptyCount = Number

        R1 = R1 + 1

        If Cells(R1, 2) = "<<empty>>" Then
            EmptyCount = EmptyCount + 1
        End If

    Loop

    'See if the bin is big enough
    Select Case PartQuantity

        Case Is <= Capacity
            Cells(R1, 2) = PartNumber
            Cells(R1, 3) = SUT
            Cells(R1, 4) = PartQuantity
            Cells(R1, 5) = Capacity
            Cells(R1, 10) = PartQuantity
            PartQuantity = 0

        Case Is > Capacity
            Cells(R1, 2) = PartNumber
            Cells(R1, 3) = SUT
            Cells(R1, 4) = Capacity
            Cells(R1, 5) = Capacity
            Cells(R1, 10) = PartQuantity
            PartQuantity = PartQuantity - Capacity
            'The number of empty bins have
            decreased by 1
            EmptyArray(Counter) =
            EmptyArray(Counter) - 1
```

```
                        End Select

                    End If
                Lowerbound = EmptyArray(Counter)
                Number = BinV - EmptySum
                Counter = Counter + 1

                'Sum the empty bins
                EmptySum = EmptySum + EmptyArray(Counter)



                Loop

            Else
                'Select the overcapacity policy
                Select Case OvercapacityPolicy

                    'Ignore and document
                    Case Is = "Ignore"
                        Call OvercapacityPolicy1

                    'Carry over to the next day
                    Case Is = "NextDay"
                        Call OvercapacityPolicy2

                    'Carry over to the next section
                    Case Is = "NextSection"
                        Call OverCapacityPolicy3

                End Select

            End If
        End If

Worksheets("CAPACITY_REFERENCES").Select
RefRow = RefRow + 1
```

```
Loop

End Sub
```

“FIRST”

```
Sub EmptyBinPolicy2() 'Chronological order: The parts are placed in bin
types in chronological order i.e. H1, H2, F1, F2

Worksheets("CAPACITY_REFERENCES").Select

'From capacity referencing worksheet define variables
RefRow = 3
RefColumn = 4

Do Until IsEmpty(Cells(RefRow, 1)) Or PartQuantity = 0

    If Cells(RefRow, 1) = Section And Cells(RefRow, 3) = SUT Then


        'Do until the partquantiy is zero or bins run out
        Do Until PartQuantity = 0 Or
        IsEmpty(Worksheets("CAPACITY_REFERENCES").Cells(2, RefColumn))

            'Check if the bin type is allowed
            If Worksheets("CAPACITY_REFERENCES").Cells(RefRow,
            RefColumn) > 0 Then

              Capacity = Cells(RefRow, RefColumn)

              'Match allowable bin to specific part
             Bin = Worksheets("CAPACITY_REFERENCES").Cells(2,
            RefColumn)

              'Select Section's worksheet
              Worksheets("SECTION_" & Section).Select
```

```
 'Assign a row number where search for an empty bin has
to start
  RowBinType = Application.WorksheetFunction.Match(Bin,
Range("A:A"), 0)


 'Make sure the search begins in the row below the bin
heading
  RowBinType = RowBinType + 1


 'Look for the first available bin in the specific bin
type
      Do Until IsEmpty(Cells(RowBinType, 2)) Or
      PartQuantity = 0


          'Find empty bin
          If Cells(RowBinType, 2) = "<<empty>>" Then


              'See if bin is big enough for parts
              Select Case PartQuantity


                  'The Capacity of the bin is sufficient
                  Case Is <= Capacity
                      Cells(RowBinType, 2) = PartNumber
                      Cells(RowBinType, 3) = SUT
                      Cells(RowBinType, 4) = PartQuantity
                      Cells(RowBinType, 5) = Capacity
                      PartQuantity = 0


                  'The capacity is not sufficient
                  Case Is > Capacity
                      Cells(RowBinType, 2) = PartNumber
                      Cells(RowBinType, 3) = SUT
                      Cells(RowBinType, 4) = Capacity
                      Cells(RowBinType, 5) = Capacity
                        PartQuantity = PartQuantity -
                        Capacity


              End Select
```

```
                            End If

                        RowBinType = RowBinType + 1
                        Loop

                End If

            'If it is not allowed, go to the next bin type
            RefColumn = RefColumn + 1

            Loop
        End If
    RefRow = RefRow + 1

    Loop

    'If the section is full
    If PartQuantity > 0 Then

        'Select the overcapacity policy
        Select Case OvercapacityPolicy

            'Ignore and document
            Case Is = "Ignore"
                Call OvercapacityPolicy1

            'Carry over to the next day
            Case Is = "NextDay"
                Call OvercapacityPolicy2

            'Carry over to the next section
            Case Is = "NextSection"
                Call OverCapacityPolicy3
        End Select

    End If
End Sub
```

“PRIORITY”

```
Sub EmptyBinPolicy3() 'Bin Type Priority: First add part to a certain bin type i.e. H1
'After that section is full refer back to first available bin policy

RowBin = 3
ColumnBin = 4

Worksheets("CAPACITY_REFERENCES").Select
MatchBinR = Application.WorksheetFunction.Match(PriorityBin, Range("2:2"), 0)

Do Until IsEmpty(Cells(RowBin, 1)) Or PartQuantity = 0

    If Cells(RowBin, 1) = Section And Cells(RowBin, 3) = SUT Then

            If Cells(RowBin, MatchBinR) > 0 Then

                'First assign capacity of priority bin
                Capacity = Cells(RowBin, MatchBinR)

                Worksheets("SECTION_" & Section).Select

                'Search for Priority bin in section
                RowBinType = Application.WorksheetFunction.Match(PriorityBin, Range("A:A"), 0)

                'Make sure the search begins in the row below the bin heading
                RowBinType = RowBinType + 1

                'Look for the first available bin in the specific bin type
                Do Until IsEmpty(Cells(RowBinType, 2)) Or PartQuantity = 0

                    'Find empty bin
```

```
            If Cells(RowBinType, 2) = "<<empty>>" Then

                'See if bin is big enough for parts
                Select Case PartQuantity

                    'The Capacity of the bin is sufficient
                    Case Is <= Capacity
                        Cells(RowBinType, 2) = PartNumber
                        Cells(RowBinType, 3) = SUT
                        Cells(RowBinType, 4) = PartQuantity
                        Cells(RowBinType, 5) = Capacity
                        PartQuantity = 0

                    'The capacity is not sufficient
                    Case Is > Capacity
                        Cells(RowBinType, 2) = PartNumber
                        Cells(RowBinType, 3) = SUT
                        Cells(RowBinType, 4) = Capacity
                        Cells(RowBinType, 5) = Capacity
                        PartQuantity = PartQuantity - Capacity

                End Select

            End If

        RowBinType = RowBinType + 1
        Loop

    End If

 'If the section is full or bintype not valid, revert back to
first available
    If PartQuantity > 0 Then

        ColumnBin = 4
        Worksheets("CAPACITY_REFERENCES").Select
```

```
 Do Until PartQuantity = 0 Or IsEmpty(Cells(RowBin,
ColumnBin))

      'Check if the bin type is allowed
If Worksheets("CAPACITY_REFERENCES").Cells(RowBin,
ColumnBin) > 0 Then

          Capacity = Cells(RowBin, ColumnBin)

          'Match allowable bin to specific part
       Bin = Worksheets("CAPACITY_REFERENCES").Cells(2,
       ColumnBin)

          'Select Section's worksheet
          Worksheets("SECTION_" & Section).Select

       'Assign a row number where search for an empty bin
       has to start
          RowBinType =
          Application.WorksheetFunction.Match(Bin,
          Range("A:A"), 0)

        'Make sure the search begins in the row below the
       bin heading
          RowBinType = RowBinType + 1

        'Look for the first available bin in the specific
       bin type
             Do Until IsEmpty(Cells(RowBinType, 2)) Or
           PartQuantity = 0

                'Find empty bin
             If Cells(RowBinType, 2) = "<<empty>>" Then

                   'See if bin is big enough for parts
                Select Case PartQuantity
              'The Capacity of the bin is sufficient
                   Case Is <= Capacity

                Cells(RowBinType, 2) = PartNumber
```

```
                                        Cells(RowBinType, 3) = SUT

                                    Cells(RowBinType, 4) =
                                    PartQuantity
                                        Cells(RowBinType, 5) =
                                        Capacity
                                        PartQuantity = 0


                                      'The capacity is not sufficient
                                      Case Is > Capacity

                                    Cells(RowBinType, 2) = PartNumber
                                        Cells(RowBinType, 3) = SUT

                                    Cells(RowBinType, 4) = Capacity
                                        Cells(RowBinType, 5) =
                                    Capacity
                                        PartQuantity = PartQuantity
                                    - Capacity


                                End Select

                            End If

                        RowBinType = RowBinType + 1
                        Loop

                    End If

                ColumnBin = ColumnBin + 1

                Worksheets("CAPACITY_REFERENCES").Select

                Loop
            End If

    End If

RowBin = RowBin + 1
```

```
Loop

'If the section is full
If PartQuantity > 0 Then

    'Select the overcapacity policy
    Select Case OvercapacityPolicy

        'Ignore and document
        Case Is = "Ignore"
            Call OvercapacityPolicy1

        'Carry over to the next day
        Case Is = "NextDay"
            Call OvercapacityPolicy2

        'Carry over to the next section
        Case Is = "NextSection"
            Call OverCapacityPolicy3

    End Select

End If
End Sub
```

“SMALLEST”

```
Sub EmptyBinPolicy4()
 'Smallest bin policy: Parts added to the smallest empty bin

'This policy is developed for the purposes of optimising part placement

'Look at the capacities of all bin types
'Look at the bin with enough space that is the smallest

Dim SBin As String
Dim BinColumn As Integer
Dim RefColumn As Integer
```

```
Dim Count As Integer
Dim i As Integer
Dim RowBinType As Integer

OvercapacityPolicy = "Ignore"

Worksheets("CAPACITY_REFERENCES").Select

'From capacity referencing worksheet define variables
RefRow = 3
RefColumn = 4
i = 4
Count = 0
k = 1 'kth smallest value



Do Until IsEmpty(Cells(RefRow, 1)) Or PartQuantity = 0

    'Found part
    If Cells(RefRow, 1) = Section And Cells(RefRow, 3) = SUT Then

        k = 1
        Column = 4
        Do Until Column > 13

            If Cells(RefRow, Column) > 0 Then
                Count = Count + 1
            End If
        Column = Column + 1

        Loop

        Do Until PartQuantity = 0 Or k > Count

            'Find the smallest capacity
```

```
    SCapacity =
    WorksheetFunction.Small(Worksheets("CAPACITY_REFERENCES")
    .Range("D" & RefRow & ": M" & RefRow), k)


     'See if the PartQuantity is more or less than smallest
    capacity


'Capacity is more than partquantity
If SCapacity >= PartQuantity Then


                    BinColumn =
    Application.WorksheetFunction.Match(SCapacity,
    Worksheets("CAPACITY_REFERENCES").Range(RefRow & ":" &
    RefRow), 0)
      SBin = Worksheets("CAPACITY_REFERENCES").Cells(2,
      BinColumn)


      Worksheets("SECTION_" & Section).Select
       'Assign a row number where search for an empty bin has
    to start
      RowBinType = Application.WorksheetFunction.Match(SBin,
      Range("A:A"), 0)


                    'Make sure the search begins in the row
    below the bin heading
      RowBinType = RowBinType + 1


                            'Look for the first available
         bin in the specific bin type
        Do Until IsEmpty(Cells(RowBinType, 2)) Or
        PartQuantity = 0


            'Find empty bin
            If Cells(RowBinType, 2) = "<<empty>>" Then
                Cells(RowBinType, 2) = PartNumber
                Cells(RowBinType, 3) = SUT
                Cells(RowBinType, 4) = PartQuantity
                Cells(RowBinType, 5) = SCapacity
                PartQuantity = 0


            End If
```

```
                    RowBinType = RowBinType + 1
                    Loop
            End If

        k = k + 1

        Loop

    End If
RefRow = RefRow + 1

Worksheets("CAPACITY_REFERENCES").Select

Loop

'If the section is full
If PartQuantity > 0 Then

    'Select the overcapacity policy
    Select Case OvercapacityPolicy

        'Ignore and document
        Case Is = "Ignore"
            Call OvercapacityPolicy1

        'Carry over to the next day
        Case Is = "NextDay"
            Call OvercapacityPolicy2

        'Carry over to the next section
        Case Is = "NextSection"
            Call OverCapacityPolicy3

    End Select
End If
End Sub
```

3. SHORTAGE POLICIES

"IGNORE"

```
Sub ShortagePolicy1()

'The shortage parts have to be documented in the newly created
SHORTAGES Worksheet



    'Enter the data into the Worksheet
    Worksheets("SHORTAGES").Range("A1048576").End(xlUp).Offset(1,
0).Value = PartNumber
    Worksheets("SHORTAGES").Range("A1048576").End(xlUp).Offset(0,
1).Value = Section
    Worksheets("SHORTAGES").Range("A1048576").End(xlUp).Offset(0,
2).Value = SUT
    Worksheets("SHORTAGES").Range("A1048576").End(xlUp).Offset(0,
3).Value = PartQuantity

    'The part quantity must now be set to zero
    PartQuantity = 0



End Sub
```

"NEXT DAY"

```
Sub ShortagePolicy2()

'In this policy the user has decided that the shortages must be carried
over to the next day and then has priority

    'Enter the data into the Worksheet
    Worksheets("SHORTAGES").Range("A1048576").End(xlUp).Offset(1,
0).Value = PartNumber
    Worksheets("SHORTAGES").Range("A1048576").End(xlUp).Offset(0,
1).Value = Section
    Worksheets("SHORTAGES").Range("A1048576").End(xlUp).Offset(0,
2).Value = SUT
    Worksheets("SHORTAGES").Range("A1048576").End(xlUp).Offset(0,
3).Value = PartQuantity
```

```
    'The part quantity must now be set to zero
    PartQuantity = 0

End Sub

Sub ShortageNextDayFlow()

Dim Row As Integer
Dim PartNumber As String
Dim Section As String
Dim SUT As String
Dim PartQuantity As Integer

Worksheets("SHORTAGES").Select

Row = 3

If Not IsEmpty(Cells(3, 1)) Then

    'Go through part flow
    Do Until IsEmpty(Cells(Row, 1))

    'Get values
    PartNumber = Worksheets("SHORTAGES").Cells(Row, 1)
    Section = Worksheets("SHORTAGES").Cells(Row, 2)
    SUT = Worksheets("SHORTAGES").Cells(Row, 3)
    PartQuantity = Worksheets("SHORTAGES").Cells(Row, 4)

    'Go to the correct Section
    Worksheets("SECTION_" & Section).Select
    RowLocal = 3


    'The program now searches for the part within the section
    Do Until IsEmpty(Cells(RowLocal, 1)) Or PartQuantity = 0
```

```
        If Cells(RowLocal, 2) = PartNumber And Cells(RowLocal, 3) = SUT
        Then

            Count = Cells(RowLocal, 4)
            Capacity = Cells(RowLocal, 5)

            Select Case PartQuantity

                'Not enough parts in bin or just enough
                Case Is >= Count
                    PartQuantity = PartQuantity - Count
                    Cells(RowLocal, 2) = "<<empty>>"
                    Cells(RowLocal, 3) = " "
                    Cells(RowLocal, 4) = " "
                    Cells(RowLocal, 5) = " "
                    Cells(RowLocal, 6) = " "

                'Enough parts in bin
                Case Is < Count
                    Cells(RowLocal, 4) = Count - PartQuantity
                   Cells(RowLocal, 6) = Cells(RowLocal, 4) /
              Cells(RowLocal, 5) * 100
                    PartQuantity = 0

            End Select

        End If

    RowLocal = RowLocal + 1
    Loop

    'When there are shortages
    If PartQuantity > 0 Then

        Select Case ShortagePolicy

            'Ignore and dokument
            Case Is = "Ignore"
```

```
                Call ShortagePolicy1

            'Carry over shortages to the next day
            Case Is = "NextDay"
                Call ShortagePolicy2

        End Select
    End If

'Delete all the data in unallocated parts worksheet
Worksheets("SHORTAGES").Range("A3:D10000").Clear

End If



End Sub
```

4. OVERCAPACITY POLICIES

"IGNORE"

```
Sub OvercapacityPolicy1()

'The unallocated parts have to be documented in the newly created
UNALLOCATED PARTS Worksheet


'If OvercapacityPolicy1 = True And PartQuantity > 0 Then


    'Enter the data into the Worksheet

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(1,
0).Value = PartNumber

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
1).Value = Section

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
2).Value = SUT

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
3).Value = PartQuantity

    'The part quantity must now be set to zero
    PartQuantity = 0


'End If


End Sub
```

"NEXT DAY"

```
Sub OvercapacityPolicy2()

'This policy adds material to the warehouse the next day.
'These parts are first added before the code runs through the days
inflow


'The idea is to also create a unallocated parts worksheet.
'The parts are then added to that worksheet
'The next day the code first runs through the unallocated parts.
'After which the data is deleted


'PART 1
```

```
'Enter the data into the Worksheet

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(1,
0).Value = PartNumber

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
1).Value = Section

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
2).Value = SUT

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
3).Value = PartQuantity

    'The part quantity must now be set to zero
    PartQuantity = 0


End Sub

Sub OverCapacityNextDayFlow()

'PART 2
'In this part the code has to run through the unallocated parts first


Dim Row As Integer
Dim PartNumber As String
Dim Section As String
Dim SUT As String
Dim PartQuantity As Integer

Worksheets("UNALLOCATED_PARTS").Select

Row = 3

If Not IsEmpty(Cells(3, 1)) Then

    'Go through part flow
    Do Until IsEmpty(Cells(Row, 1))


    'Get values
    PartNumber = Worksheets("UNALLOCATED_PARTS").Cells(Row, 1)
    Section = Worksheets("UNALLOCATED_PARTS").Cells(Row, 2)
    SUT = Worksheets("UNALLOCATED_PARTS").Cells(Row, 3)
    PartQuantity = Worksheets("UNALLOCATED_PARTS").Cells(Row, 4)


    'Go to the correct Section and define variables
```

```
    Worksheets("SECTION_" & Section).Select
    RowLocal = 3

        'Find part in the section
        Do Until IsEmpty(Cells(RowLocal, 1)) Or PartQuantity = 0

            If Cells(RowLocal, 2) = PartNumber And Cells(RowLocal, 3) =
SUT Then

            Count = Cells(RowLocal, 4)
            Capacity = Cells(RowLocal, 5)
            AvailableSpace = Capacity - Count

                'Check if there is space in bin
                Select Case AvailableSpace

                    'Not enough space available in bin
                    Case Is < PartQuantity
                        PartQuantity = PartQuantity - AvailableSpace
                        Cells(RowLocal, 10) = AvailableSpace
                        Cells(RowLocal, 4) = Capacity
                        Cells(RowLocal, 6) = 100


                    'Enough space available in bin
                    Case Is >= PartQuantity
                        Cells(RowLocal, 4) = Count + PartQuantity
                        Cells(RowLocal, 10) = PartQuantity
                        Cells(RowLocal, 6) = (Count + PartQuantity) /
Capacity * 100
                        PartQuantity = 0


                End Select

            End If

        RowLocal = RowLocal + 1

        Loop

    Row = Row + 1
    Loop

    'If there are still parts left
    If PartQuantity > 0 Then

        Select Case EmptyBinPolicy

            Case Is = "Random"

                'Place parts randomly
```

```
                Call EmptyBinPolicy1

            Case Is = "First"
                'Bin Type Priority
                Call EmptyBinPolicy2

            Case Is = "Priority"
                'Smallest bin
                Call EmptyBinPolicy3

            Case Is = "Smallest"
                'First available bin
                Call EmptyBinPolicy4

        End Select

    End If

'Delete all the data in unallocated parts worksheet
Worksheets("UNALLOCATED_PARTS").Range("A3:D10000").Clear

End If
```

"NEXT SECTION"

```
Sub OverCapacityPolicy3()

'In this policy the user chooses to add materials to the next section
if a particular section is full


'After the Part searches for space in a section and no space is found
the part will proceed to start searching in the next section

MatchValue = Application.WorksheetFunction.Match(Section, SectionArray,
0)

Select Case MatchValue


    'If it is not the last section in the array and not the first
    Case 2 To SectionCount - 1


        'First go through section after section
        i = MatchValue

        'Go through the entire array from section to finish
```

```
        Do Until i = SectionCount + 1 Or PartQuantity = 0

            Section = SectionArray(i)

            'Go to the correct Section and define variables
            Worksheets("SECTION_" & Section).Select
            RowLocal = 3

            'Find part in the section
            Do Until IsEmpty(Cells(RowLocal, 1)) Or PartQuantity = 0

                If Cells(RowLocal, 2) = PartNumber And Cells(RowLocal,
3) = SUT Then

                Count = Cells(RowLocal, 4)
                Capacity = Cells(RowLocal, 5)
                AvailableSpace = Capacity - Count

                    'Check if there is space in bin
                    Select Case AvailableSpace

                        'Not enough space available in bin
                        Case Is < PartQuantity
                            'Record in the unallocated parts worksheet

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(1,
0).Value = PartNumber

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
1).Value = Section

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
2).Value = SUT

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
3).Value = PartQuantity

                            Worksheets("SECTION_" & Section).Select
                            PartQuantity = PartQuantity -
AvailableSpace
                            Cells(RowLocal, 10) = AvailableSpace
                            Cells(RowLocal, 4) = Capacity
                            Cells(RowLocal, 6) = 100


                        'Enough space available in bin
                        Case Is >= PartQuantity
                            'Record in the unallocated parts worksheet

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(1,
0).Value = PartNumber
```

```
Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
1).Value = Section

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
2).Value = SUT

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
3).Value = PartQuantity

                            Worksheets("SECTION_" & Section).Select
                            Cells(RowLocal, 4) = Count + PartQuantity
                            Cells(RowLocal, 10) = PartQuantity
                            Cells(RowLocal, 6) = (Count + PartQuantity)
/ Capacity * 100
                            PartQuantity = 0


                    End Select

                End If

            RowLocal = RowLocal + 1

            Loop

            'If there are still parts left
            If PartQuantity > 0 Then

                Select Case EmptyBinPolicy

                    Case Is = "Random"
                        'Place parts randomly
                        Call EmptyBinPolicy1

                    Case Is = "First"
                        'Bin Type Priority
                        Call EmptyBinPolicy2

                    Case Is = "Priority"
                        'Smallest bin
                        Call EmptyBinPolicy3

                    Case Is = "Smallest"
                        'First available bin
                        Call EmptyBinPolicy4

                End Select
```

```
            End If


        i = i + 1
        Loop

        'Second go through sections before section
        i = 1

        'Go through the entire array from section to finish
        Do Until i = MatchValue Or PartQuantity = 0

            Section = SectionArray(i)

            'Go to the correct Section and define variables
            Worksheets("SECTION_" & Section).Select
            RowLocal = 3

            'Find part in the section
            Do Until IsEmpty(Cells(RowLocal, 1)) Or PartQuantity = 0

                If Cells(RowLocal, 2) = PartNumber And Cells(RowLocal,
3) = SUT Then

                Count = Cells(RowLocal, 4)
                Capacity = Cells(RowLocal, 5)
                AvailableSpace = Capacity - Count

                    'Check if there is space in bin
                    Select Case AvailableSpace


                        'Not enough space available in bin
                        Case Is < PartQuantity
                            'Record in the unallocated parts worksheet

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(1,
0).Value = PartNumber

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
1).Value = Section

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
2).Value = SUT

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
3).Value = PartQuantity

                            Worksheets("SECTION_" & Section).Select
```

```
                            PartQuantity = PartQuantity -
AvailableSpace
                            Cells(RowLocal, 10) = AvailableSpace
                            Cells(RowLocal, 4) = Capacity
                            Cells(RowLocal, 6) = 100

                        'Enough space available in bin
                        Case Is >= PartQuantity
                            'Record in the unallocated parts worksheet

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(1,
0).Value = PartNumber

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
1).Value = Section

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
2).Value = SUT

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
3).Value = PartQuantity

                            Worksheets("SECTION_" & Section).Select
                            Cells(RowLocal, 4) = Count + PartQuantity
                            Cells(RowLocal, 10) = PartQuantity
                            Cells(RowLocal, 6) = (Count + PartQuantity)
/ Capacity * 100
                            PartQuantity = 0


                    End Select

                End If

            RowLocal = RowLocal + 1

            Loop

            'If there are still parts left
            If PartQuantity > 0 Then

                Select Case EmptyBinPolicy

                    Case Is = "Random"
                        'Place parts randomly
                        Call EmptyBinPolicy1

                    Case Is = "First"
                        'Bin Type Priority
                        Call EmptyBinPolicy2

                    Case Is = "Priority"
                        'Smallest bin
```

```
                        Call EmptyBinPolicy3

                    Case Is = "Smallest"
                        'First available bin
                        Call EmptyBinPolicy4

                End Select

            End If

        i = i + 1
        Loop

    'If it is the first
    Case Is = 1

        i = 2 'Start from the second section

        'Go through the entire array from start to finish
        Do Until i = SectionCount + 1 Or PartQuantity = 0

            Section = SectionArray(i)

            'Go to the correct Section and define variables
            Worksheets("SECTION_" & Section).Select
            RowLocal = 3

            'Find part in the section
            Do Until IsEmpty(Cells(RowLocal, 1)) Or PartQuantity = 0

                If Cells(RowLocal, 2) = PartNumber And Cells(RowLocal,
3) = SUT Then

                Count = Cells(RowLocal, 4)
                Capacity = Cells(RowLocal, 5)
                AvailableSpace = Capacity - Count

                    'Check if there is space in bin
                    Select Case AvailableSpace

                        'Not enough space available in bin
                        Case Is < PartQuantity
                            'Record in the unallocated parts worksheet

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(1,
0).Value = PartNumber
```

```
Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
1).Value = Section

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
2).Value = SUT

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
3).Value = PartQuantity

                            Worksheets("SECTION_" & Section).Select
                            PartQuantity = PartQuantity -
AvailableSpace
                            Cells(RowLocal, 10) = AvailableSpace
                            Cells(RowLocal, 4) = Capacity
                            Cells(RowLocal, 6) = 100


                        'Enough space available in bin
                        Case Is >= PartQuantity
                            'Record in the unallocated parts worksheet

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(1,
0).Value = PartNumber

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
1).Value = Section

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
2).Value = SUT

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
3).Value = PartQuantity

                            Worksheets("SECTION_" & Section).Select
                            Cells(RowLocal, 4) = Count + PartQuantity
                            Cells(RowLocal, 10) = PartQuantity
                            Cells(RowLocal, 6) = (Count + PartQuantity)
/ Capacity * 100
                            PartQuantity = 0


                    End Select

                End If

            RowLocal = RowLocal + 1

            Loop

            'If there are still parts left
            If PartQuantity > 0 Then
```

```
                Select Case EmptyBinPolicy

                    Case Is = "Random"
                        'Place parts randomly
                        Call EmptyBinPolicy1

                    Case Is = "First"
                        'Bin Type Priority
                        Call EmptyBinPolicy2

                    Case Is = "Priority"
                        'Smallest bin
                        Call EmptyBinPolicy3


                    Case Is = "Smallest"
                        'First available bin
                        Call EmptyBinPolicy4

                End Select

            End If

        i = i + 1
        Loop

    'If it is the last
    Case Is = SectionCount

        i = 1 'Start from the beginning to the second last section


        'Go through the entire array from start to finish
        Do Until i = SectionCount Or PartQuantity = 0

            Section = SectionArray(i)

            'Go to the correct Section and define variables
            Worksheets("SECTION_" & Section).Select
            RowLocal = 3


            'Find part in the section
            Do Until IsEmpty(Cells(RowLocal, 1)) Or PartQuantity = 0

                If Cells(RowLocal, 2) = PartNumber And Cells(RowLocal,
3) = SUT Then

                Count = Cells(RowLocal, 4)
```

```
                    Capacity = Cells(RowLocal, 5)
                    AvailableSpace = Capacity - Count

                        'Check if there is space in bin
                        Select Case AvailableSpace


                            'Not enough space available in bin
                            Case Is < PartQuantity
                                'Record in the unallocated parts worksheet

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(1,
0).Value = PartNumber

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
1).Value = Section

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
2).Value = SUT

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
3).Value = PartQuantity

                                Worksheets("SECTION_" & Section).Select
                                PartQuantity = PartQuantity -
AvailableSpace
                                Cells(RowLocal, 10) = AvailableSpace
                                Cells(RowLocal, 4) = Capacity
                                Cells(RowLocal, 6) = 100

                            'Enough space available in bin
                            Case Is >= PartQuantity
                                'Record in the unallocated parts worksheet

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(1,
0).Value = PartNumber

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
1).Value = Section

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
2).Value = SUT

Worksheets("UNALLOCATED_PARTS").Range("A1048576").End(xlUp).Offset(0,
3).Value = PartQuantity

                                Worksheets("SECTION_" & Section).Select
                                Cells(RowLocal, 4) = Count + PartQuantity
                                Cells(RowLocal, 10) = PartQuantity
                                Cells(RowLocal, 6) = (Count + PartQuantity)
/ Capacity * 100
                                PartQuantity = 0
```

```
                    End Select

                End If

            RowLocal = RowLocal + 1

            Loop

        i = i + 1
        Loop

End Select




End Sub

Sub OverCapacityNextSectionOutflow()

'If the next section policy is chosen the parts first has to be
extracted from that section
'Check if the part has been added to another section

Worksheets("UNALLOCATED_PARTS").Select

RowSearch = 3

Do Until IsEmpty(Cells(RowSearch, 1)) Or PartQuantity = 0

    'Part found
    If Cells(RowSearch, 1) = PartNumber And Cells(RowSearch, 3) = SUT
Then

        'New Section
        Section2 = Cells(RowSearch, 2)

        'Go to that section
        Worksheets("SECTION_" & Section2).Select
        RowLocal = 3

            'The program now searches for the part within the section
            Do Until IsEmpty(Cells(RowLocal, 1)) Or PartQuantity = 0

                If Cells(RowLocal, 2) = PartNumber And Cells(RowLocal,
3) = SUT Then

                Count = Cells(RowLocal, 4)
                Capacity = Cells(RowLocal, 5)

                    Select Case PartQuantity

                        'Not enough parts in bin or just enough
```

```
                        Case Is >= Count
                            PartQuantity = PartQuantity - Count
                            Cells(RowLocal, 2) = "<<empty>>"
                            Cells(RowLocal, 3) = " "
                            Cells(RowLocal, 4) = " "
                            Cells(RowLocal, 5) = " "
                            Cells(RowLocal, 6) = " "

                            'Delete the record

Worksheets("UNALLOCATED_PARTS").Range(RowSearch & ":" &
RowSearch).Select
                            Selection.Delete Shift:=xlUp

                        'Enough parts in bin
                        Case Is < Count
                            Cells(RowLocal, 4) = Count - PartQuantity
                            Cells(RowLocal, 6) = Cells(RowLocal, 4) /
Cells(RowLocal, 5) * 100
                            PartQuantity = 0


                    End Select

                End If

            RowLocal = RowLocal + 1
            Loop

    End If

Worksheets("UNALLOCATED_PARTS").Select

RowSearh = RowSearch + 1
Loop



End Sub
```

5. CAPACITY FLUCTUATION AND DATA DATA

```
Sub DataTables()

Dim SectionArray() As String
Dim BinArray() As String
Dim Elements As Integer
Dim i As Integer
Dim Row As Integer

Elements = 6

ReDim SectionArray(1 To 6)

SectionArray(1) = "DBK"
SectionArray(2) = "DTM"
SectionArray(3) = "DBS"
SectionArray(4) = "DLO"
SectionArray(5) = "DUS"
SectionArray(6) = "DNS"

Elements = Elements - 1


Row = 3
k = 1

SimDays = 3

Do Until k > 6

    Worksheets("DATA").Cells(Row, 1) = SectionArray(k)
    Worksheets("DATA").Cells(Row + 1, 1) = "Empty"
    Worksheets("DATA").Cells(Row + 2, 1) = "Partially Full"
    Worksheets("DATA").Cells(Row + 3, 1) = "Full"

    DaysColumn = 2
    i = 1
    'Now add simulation days in and outflow
    Do Until i = SimDays + 1
        Worksheets("DATA").Cells(Row, DaysColumn) = "Day" & i & " " &
"In"
        Worksheets("DATA").Cells(Row, DaysColumn + 1) = "Day" & i & " "
& "Out"
    DaysColumn = DaysColumn + 2
    i = i + 1
    Loop
Row = Row + 5
k = k + 1
Loop

'Now add the heading for the bins and define the Binarray
Col = 4 'Column
```

```
n = 1
BinTypeCount = 0

'Count bintypes
Do Until IsEmpty(Worksheets("CAPACITY_REFERENCES").Cells(2, Col))
    BinTypeCount = BinTypeCount + 1
Col = Col + 1
Loop

'ReDim array
ReDim BinArray(1 To BinTypeCount)

Col = 4

SimDays = 3

Do Until IsEmpty(Worksheets("CAPACITY_REFERENCES").Cells(2, Col))

    BinArray(n) = Worksheets("CAPACITY_REFERENCES").Cells(2, Col)

    Worksheets("DATA").Cells(Row, 1) = BinArray(n)
    Worksheets("DATA").Cells(Row + 1, 1) = "Empty"
    Worksheets("DATA").Cells(Row + 2, 1) = "Partially Full"
    Worksheets("DATA").Cells(Row + 3, 1) = "Full"

    DaysColumn = 2
    i = 1
    'Now add simulation days in and outflow
    Do Until i = SimDays + 1
        Worksheets("DATA").Cells(Row, DaysColumn) = "Day" & i & " " &
"In"
        Worksheets("DATA").Cells(Row, DaysColumn + 1) = "Day" & i & " "
& "Out"
    DaysColumn = DaysColumn + 2
    i = i + 1
    Loop
n = n + 1
Row = Row + 5
Col = Col + 1
Loop

End Sub
```

FLUCTUATION

```
Sub CapacityFluctuations()
'This output module measures the fluctuations in capacity in each
section, bintype and warehouse


Dim RowSearch As Integer
Dim BinType As String
Dim R As Integer
Dim CountE As Integer
Dim CountP As Integer
```

```
Dim CountF As Integer
Dim EmptyS1 As Integer
Dim PartialS1 As Integer
Dim FullS1 As Integer
Dim SectionName As String
Dim Space As Integer
Dim SectionArray(1 To 6) As String
Dim BinArray(1 To 10) As String


SectionArray(1) = "DBK"
SectionArray(2) = "DTM"
SectionArray(3) = "DBS"
SectionArray(4) = "DLO"
SectionArray(5) = "DUS"
SectionArray(6) = "DNS"


BinArray(1) = "H1"
BinArray(2) = "H2"
BinArray(3) = "H3"
BinArray(4) = "H4"
BinArray(5) = "N1"
BinArray(6) = "NE"
BinArray(7) = "BK"
BinArray(8) = "F1"
BinArray(9) = "F2"
BinArray(10) = "F3"




DataRow = 4
SectionCount = 6
i = 1

'Go through all sections
Do Until i > SectionCount

    'Retrieve section name
    SectionName = SectionArray(i)

    Debug.Print SectionName

    RowBins = 2
    RowSearch = 2

    'Select Worksheet
    Worksheets("SECTION_" & SectionName).Select

    EmptyS = 0
    PartialS = 0
    FullS = 0
```

```
'Go through entire section
Do Until IsEmpty(Cells(RowSearch, 1))

    CountE = 0
    CountP = 0
    CountF = 0
    BinType = Cells(RowSearch, 1)
    RowSearch = RowSearch + 1


    'Go through one bintype
    Do Until IsEmpty(Cells(RowSearch, 2))


        'Count all the empty, partially and full bins
        If IsEmpty(Cells(RowSearch, 6)) Then
            CountE = CountE + 1

        Else
            PercentageFull = Cells(RowSearch, 6)
            Select Case PercentageFull

                'Partially full
                Case 0.001 To 99
                    CountP = CountP + 1

                'Full bins
                Case Is = 100
                    CountF = CountF + 1

            End Select
        End If

    RowSearch = RowSearch + 1


    Loop
    'Record bintype empty, partially full and full
    Cells(RowBins, 12) = BinType
    Cells(RowBins, 13) = CountE
    Cells(RowBins, 14) = CountP
    Cells(RowBins, 15) = CountF
    RowBins = RowBins + 1

'Calculate all empty, partial and full in entire section
EmptyS = EmptyS + CountE
PartialS = PartialS + CountP
FullS = FullS + CountF

Loop

'Add data
Worksheets("DATA").Cells(DataRow, DataColumn) = EmptyS
Worksheets("DATA").Cells(DataRow + 1, DataColumn) = PartialS
```

```
    Worksheets("DATA").Cells(DataRow + 2, DataColumn) = FullS


DataRow = DataRow + 5
i = i + 1
Loop

'Set value for counter i, R and j
i = 1
R = 1
BR = 21 'BinRow
BR3 = 0
BinTypeCount = 10

'Create a graph for each bintype
Do Until i > BinTypeCount

    Bin = BinArray(i)

    Debug.Print Bin

    j = 1
    EmptyBT = 0
    PartialBT = 0
    FullBT = 0

    'Go through all section to add the bintype data
    Do Until j > SectionCount

        SectionName = SectionArray(j)

        Debug.Print SectionName

        Worksheets("SECTION_" & SectionName).Select

        'If the bintype is found in the section

        'Not found
        If Application.WorksheetFunction.CountIf(Range("L:L"), Bin) = 0
Then
            EmptyB = 0
            PartialB = 0
            FullB = 0

        'Found
        Else
            Row = Application.WorksheetFunction.Match(Bin,
Range("L:L"), 0)
            EmptyB = Cells(Row, 13)
            PartialB = Cells(Row, 14)
            FullB = Cells(Row, 15)
        End If

    EmptyBT = EmptyBT + EmptyB
```

```
    PartialBT = PartialBT + PartialB
    FullBT = FullBT + FullB
    j = j + 1
    Loop

    'Go to data worksheet
    Worksheets("DATA").Select

    RowB = Application.WorksheetFunction.Match(Bin, Range("A:A"), 0)

    'Record values in worksheet
    Cells(RowB + 1, DataColumn) = EmptyBT
    Cells(RowB + 2, DataColumn) = PartialBT
    Cells(RowB + 3, DataColumn) = FullBT


i = i + 1

Loop

End Sub
```

CHARTS

```
Sub CreateCharts()

'When the model is finished the charts are created

Dim SectionArray(1 To 6) As String
Dim BinArray(1 To 10) As String


SimDays = 3

SectionArray(1) = "DBK"
SectionArray(2) = "DTM"
SectionArray(3) = "DBS"
SectionArray(4) = "DLO"
SectionArray(5) = "DUS"
SectionArray(6) = "DNS"

BinArray(1) = "H1"
BinArray(2) = "H2"
BinArray(3) = "H3"
BinArray(4) = "H4"
BinArray(5) = "N1"
BinArray(6) = "NE"
BinArray(7) = "BK"
BinArray(8) = "F1"
BinArray(9) = "F2"
BinArray(10) = "F3"

SpaceT = 0
Column = 2 * SimDays + 1
```

```
Row = 3
i = 1
SectionCount = 6

'First create the section graphs
Do Until i > SectionCount

Worksheets("OUTPUT").Select

    With ActiveSheet.ChartObjects.Add(Left:=50, Width:=375, Top:=SpaceT
+ 50, Height:=225)
    .Chart.SetSourceData Source:=Sheets("DATA").Range("A" & Row & ":G"
& Row + 3)
    .Chart.ChartType = xlColumnClustered
    .Chart.HasTitle = True
    .Chart.ChartTitle.Text = "Section" & SectionArray(i)
    .Chart.SetElement (msoElementChartTitleAboveChart)
    End With

SpaceT = SpaceT + 275
Row = Row + 5
i = i + 1
Loop

SpaceT = 0
j = 1
BinTypeCount = 10

'Create bin graphs
Do Until j > BinTypeCount

Worksheets("OUTPUT").Select

    With ActiveSheet.ChartObjects.Add(Left:=475, Width:=375,
Top:=SpaceT + 50, Height:=225)
    .Chart.SetSourceData Source:=Sheets("DATA").Range("A" & Row & ":G"
& Row + 3)
    .Chart.ChartType = xlColumnClustered
    .Chart.HasTitle = True
    .Chart.ChartTitle.Text = BinArray(j) & "Bin"
    .Chart.SetElement (msoElementChartTitleAboveChart)
    End With

SpaceT = SpaceT + 275
Row = Row + 5
j = j + 1
Loop

End Sub
```