Coal-to-Liquid logistics at Sasol

Project Leader

Professor Adendorff

Student

Jakus van Rooyen

25245679

FACULTY OF ENGINEERING, BUILT ENVIRONMENT AND INFORMATION TECHNOLOGY
UNIVERSITY OF PRETORIA

October 2010

Executive Summary

Sasol, the leading manufacturer of synthetic fuels from coal in the world, has embarked on setting up a number of Coal-to-Liquid (CTL) Joint Ventures (JV's) globally. Site selection remains one of the most important drivers for setting up a JV internationally. A number of logistics chain factors can influence this decision-making process. The development of a programming model that can analyse logistics modes and determine logistics costs of a possible CTL location is important. This study aims to determine the cost types associated to carrying specific product types and volumes between locations and the selection of the most suitable logistics modes. This will aid in the selection of an optimal CTL site.

Table of Contents

Executive Summary i
Table of Contents ii
1. Introduction and Background 1
1.1 Sasol 1
1.2 Sasol Synfuels International (SSI) division 1
1.3 Coal-to-Liquid (CTL) vs. Gas-to-Liquid (GTL) 1
1.4 Coal-to-Liquid (CTL) logistics 1
2. Identification of the problem 2
2.1 Introduction to problem 2
2.2 The basic logistics problem 3
2.3 Current project environment 3
2.3.1 Objective type and number of objectives 3
2.3.2 Planning horizon 4
2.3.3 Period planning 4
2.3.4 Solution space 4
2.3.5 Logistics modes 4
2.3.7 Number of facility locations 4
2.4 Other assumptions: certainties, risks and uncertainties [7] 5
2.4.1 Deterministic components for the first tier 5
2.4.2 Stochastic components (risks and uncertainties) for the second tier 5
2.5.3 Components that will have no impact on the final solution 5
3. Make a hypothesis 7
3.1 First Tier Project Scope (Deterministic Model) 7
3.2 Second Tier Project Scope (Probabilistic elements added) 8
3.3 Assumptions 9
3.4 Solution Types [4] 9
3.4.1 A heuristic solution and exact solution 9
3.4.2 A specific algorithm and general solver 9
3.5 Application to problem 9
3.5.1 Solution type requirements 9
3.6 Supplementary methods and methodologies 10
3.6.1 The seven steps of the scientific method (Context) 10
3.6.2 Loose i2 programming method (Content) 10
3.6.3 Spiral model for reporting 11
4. Create a solution 12
4.1 Typical conceptual solution to current environment 12
4.1.1 Mathematical formulation example [9] 12
4.2 New improved conceptual solutions 13
4.2.1 Mathematical formulation for tier one 13
4.2.2 Notes 13
4.2.3 Improvements in new conceptual solution 13
4.2.4 Mathematical formulation for tier two 14
4.2.5 Notes: 16
4.2.6 Improvements over tier one solution 16
5. Execution of solution 17
5.1 Basic information 17
5.1.1 Execution of solution summary 17
5.1.2 Programming Language 17

5.1.3 Programming philosophy 18
5.2 Process of program execution: 19
5.2.1 Main execution 19
5.2.2 Input data: 2.1 19
5.2.3 Travel information: 9.1 19
5.2.4 Calculate logistics costs: 4.1 19
5.3 Program Description 20
5.3.1 Open Program 20
5.3.2 Inputs 22
5.3.3 Storage of data 31
5.3.4 Outputs 33
5.3.5 Limitations 37
5.4 UML class diagrams, concise class description, Java code 38
Main Class 38
Program Run Classes 38
Input Classes 41
Utility classes 50
Calculate and output class 53
8. Analyse data 59
8.1 Data Corruption 59
8.1.1 Class interface 59
8.1.2 Textbox to variable 59
8.1.3 Types of data 59
8.2 File Saving 59
8.2.1 Incomplete dataset 59
8.2.2 Record to file 59
8.3 File and variable management 59
8.3.1 Variables has been define incorrectly 59
8.3.2 Wrong input programming heuristic 59
8.4 Incomplete output 60
8.4.1 Not all data shown 60
8.4.2 Output structure 60
8.4.3 Calculations done incorrectly 60
8.4.4 Programming done wrong 60
9. Go over solution 61
9.1.1 Mathematical verification 61
10. Make a conclusion 62
11. Supplements 63
11.1 Appendix 1 (Definitions) 63
11.2 Appendix 2 (References) 64

1. Introduction and Background

1.1 Sasol

Sasol was established in 1950 to produce synthetic fuels from coal for the South African market. South Africa does not have a known viable crude oil reserve of its own. Since 1950, Sasol has provided unique, innovative, and competitive technology solutions to add value to coal, oil and gas reserves by transforming these raw materials into liquid fuel, fuel components and chemicals. Sasol markets directly to customers in 30 countries.

1.2 Sasol Synfuels International (SSI) division

Sasol Synfuels International (SSI) and Sasol Petroleum manage Sasol's international ventures. SSI is concerned with Gas-to-Liquid (GTL) and Coal-to-Liquid (CTL) projects abroad.

1.3 Coal-to-Liquid (CTL) vs. Gas-to-Liquid (GTL)

Coal-to-Liquid (CTL) is the process of converting low grade coal into fuel, fuel components and other chemicals. Opposed to this is Sasol's Gas-to-Liquid (GTL) process, which uses natural gas instead of coal as feedstock.

Sasol operates commercial plants of both types globally and is investigating additional opportunities, at various international locations.

1.4 Coal-to-Liquid (CTL) logistics

Site selection remains one of the importation drivers in setting up a Coal-to-Liquid (CTL) Joint Venture (JV). A number of logistics chain factors can influence this decision-making process. The focus of this project is a programming model that can evaluate feasible CTL sites by selecting of the most suitable logistics and determining logistics costs for all possible sites.

2. Identification of the problem

2.1 Introduction to problem

Sasol frequently needs to decide between multiple locations for its CTL plants. This decision is based upon multiple criterion, the most important being the nature and quality of the coal reserve. After elimination of sites which do not pass coal reserve size and quality requirements, the sites are evaluated based on additional criteria.

These criteria include determining the capital and operational cost associated with establishing and operating a functional logistics chain for each feasible location. Costs are driven by material logistics between transfer points.

The aim of this project is to develop an algorithmic programming method that can be used to determine the logistics chain costs associated with a specific CTL site. This will be used to evaluate which site presents the greatest opportunity in terms of logistics chain cost saving.

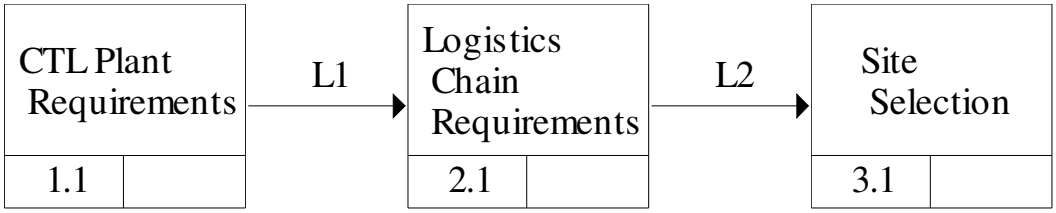


Figure 1: Project aims

2.2 The basic logistics problem

Facility location selection within a logistics chain already is a well established field within the research environment. The problem is presented in the form of selecting a facility location within a given space and servicing customer demand [3]. It is assumed that the setup costs are fixed and do not vary between potential sites. [1, 2]

2.3 Current project environment

2.3.1 Objective type and number of objectives

Cost and/or profits are the two types of objective classes within a logistics chain. The conclusion of this project must thus be in one of these types of measurements.

Other diverging objectives with a strong correlation to cost and/or profits might also be used in analysis of sophisticated logistics chains. Examples of these multiple objective function types include logistics chain utility and mean outcome [3, 11].

A survey that has been done, counting the number of times a performance measurement class has been used, suggests that cost is foremost in use in the objective function of a logistics chain [4]:

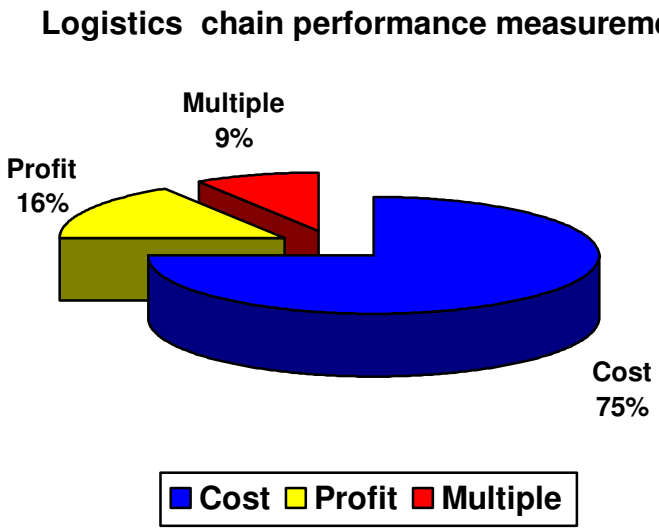


Figure 5: Performance measurements

Assumption: The main objective will be cost saving. The second tier will include a multifaceted study that also involves risk and storage capacity assessment and will be a juxtaposition measurement of only cost saving within the logistics chain.

2.3.2 Planning horizon

Three planning horizons can be distinguished in a logistics chain: strategic, tactical and operational [5]. These planning horizon decisions are time-dependent and have a strong association with the types of decision made. Strategic planning has a long lasting, permanent effect on the logistics chain, while operational planning is short-term (normally less then a week), such as meeting demand on time.

Assumption: This study is done at the strategic level.

2.3.3 Period planning

Yet another time-dependent decision criterion lies with a single period as opposed to a multiple period timeframe. Where a single period assumes that the quantities of materials that are moved between departments remain constant, while in multiple period logistics design, it is assumes that the supply and demand will vary in a predictable way over time. Both of these modes have their drawbacks. With the single period assumption, the rigidity both makes it stable and inflexible, while with multiple periods, projections are not always accurate. Approximately 82% of logistics chain research publications deal only with single period planning [4]. [6]

Assumption: Single period planning.

2.3.4 Solution space

In a discrete solution space only predefined, limited and feasible locations will be evaluated. A continuous solution space differs in the available locations, which are an infinite number of points. An optimum location is theoretically possible, and is always available. This optimal location is for a given set of criterion. Both of these solution spaces are two dimensional in space.

Assumption: Discrete solution space in 2 dimensions.

2.3.5 Logistics modes

Most studies assume that only one type of logistics method is used, normally that mode is road. Unfortunately, this will not be useful for the logistics chain of a CTL plant, as there are a number of logistics modes available for the transportation of materials.

Assumption: Logistics evaluation will include:

- *Waterway*
- *Rail*
- *Road*
- *Pipeline*
- Conveyer

2.3.7 Number of facility locations

A facility location is a point where the CTL plant can be located within the logistics chain. The CTL plant is the only facility that can be positioned. All other facilities are fixed to a spot.

Assumption: The one and only facility that can be positioned is the CTL plant.

2.4 Other assumptions: certainties, risks and uncertainties [7]

2.4.1 Deterministic components for the first tier

- Inputs:
  - Demand from departments
  - Distances between departments
  - Costs (capex and opex) of logistics modes
  - Infrastructure availability and capacity
- Outputs:
  - Logistics costs in NPV of CTL plants at varies discrete locations

2.4.2 Stochastic components (risks and uncertainties) for the second tier

- Inputs:
  - Reliability and robustness of supply
  - Lead-time
  - Parcel sizes
  - Logistics modes capacities and availabilities
- Outputs:
  - Risk Impact on logistics chain
  - Risk pooling
  - Storage capacity requirements

2.5.3 Components that will have no impact on the final solution

- Reverse logistics
- Procurement
- Production
- Routing
- Other financial factors such as: internal factors, incentives and budget constraints
- Relocations
- Bill of materials

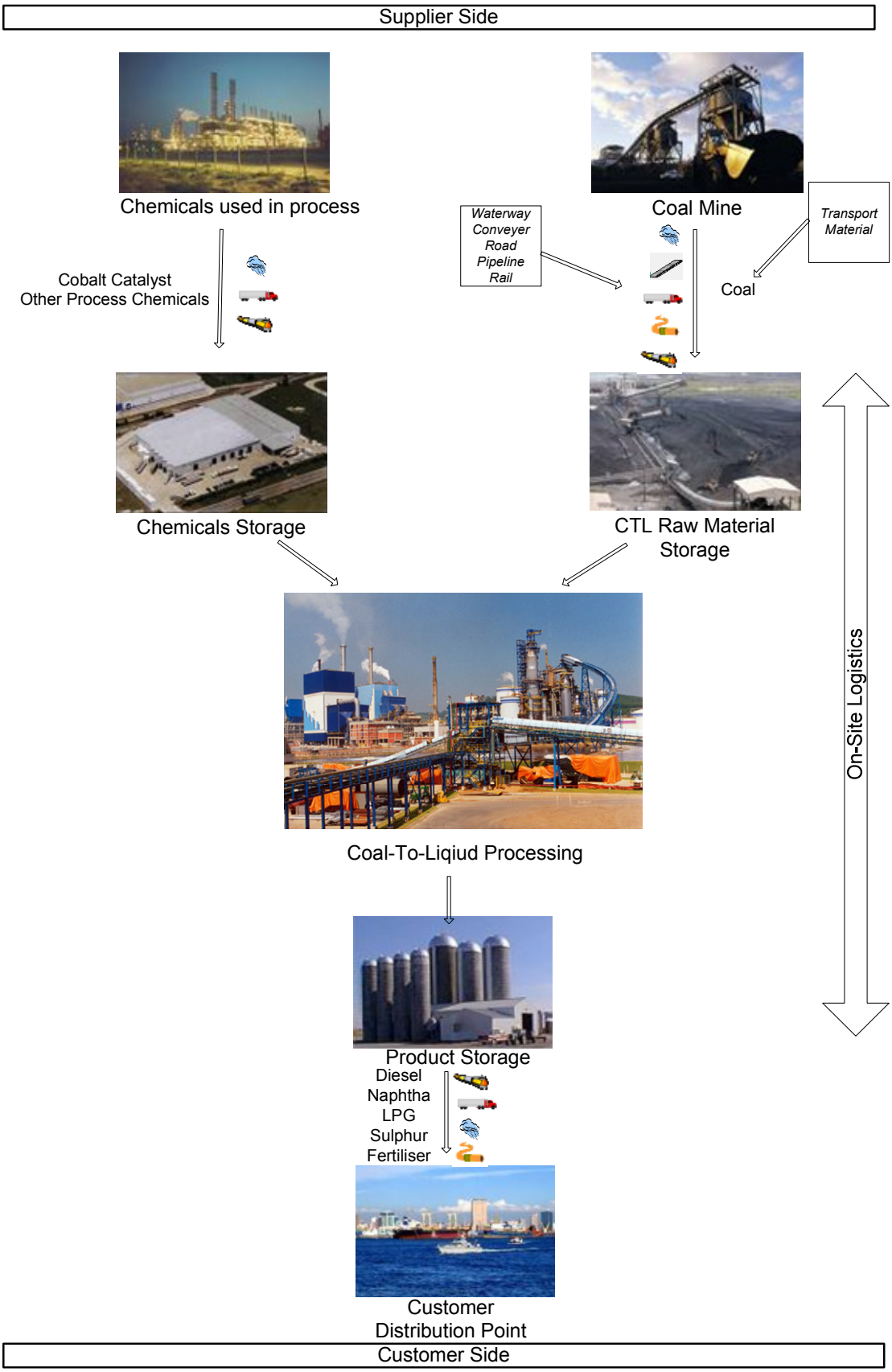


Figure 2: Typical CTL logistics chain

3. Make a hypothesis

3.1 First Tier Project Scope (Deterministic Model)

A deterministic model will be constructed to calculate the logistics chain costs associated with a specific CTL site. This will use a predefined set of in-bound and out-bound logistics requirements. The following materials will be included:

- In-bound:
  - Coal
  - Cobalt Catalyst
  - Other Process Chemicals
- Out-bound
  - Diesel
  - Naphtha
  - LPG
  - Sulphur
  - Fertiliser

These materials will be used to evaluate the following five logistics modes:

- Road
- Rail
- Conveyer
- Pipeline
- Waterway

The cost of these five different logistics modes will be defined within a certain confidence interval through the use of capex (Capital Expenditure) and opex (Operational Expenditure) for each of the materials and the distance of travel. Sasol utilises existing infrastructure when available, but may need to construct and maintain new infrastructure. The model needs to cater for both scenarios.

Site selection will take place to minimise the logistics chain costs.

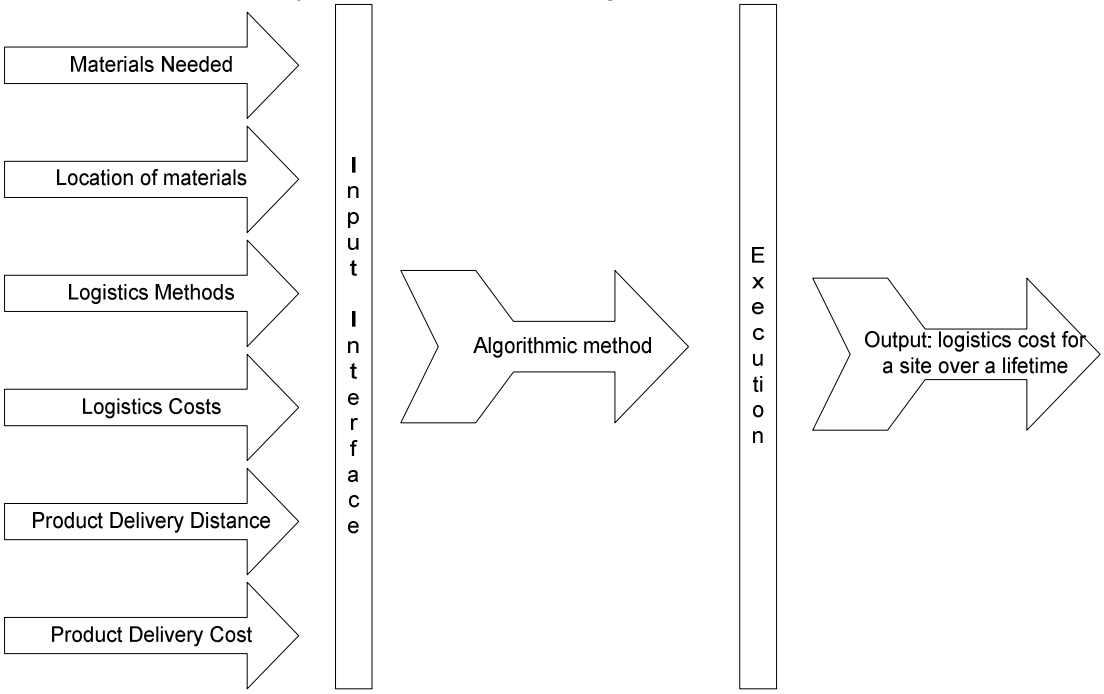


Figure 3: First tier project scope diagram

3.2 Second Tier Project Scope (Probabilistic elements added)

This project scope will place emphasis on the variations within the logistics chain that might be the cause of poor performance.

Certain logistics modes have limited capacity and/or minimum capacity, so the use of more than a single logistics method for every material is required. Certain logistics modes might also be impractical for some materials and areas. Also, parcel sizes will be examined together with the impact it will have on costs associated with the logistics chain.

The lead-time and reliability of logistics and other causes of variance will be used to give an indicator of the impact certain events will have on the logistics chain and on the storage capacity of the plant. A concise evaluation of storage capacity for all materials in the scope will also be done.

The use of multiple mines to feed the CTL plant and more than one customer distribution points will be examined.

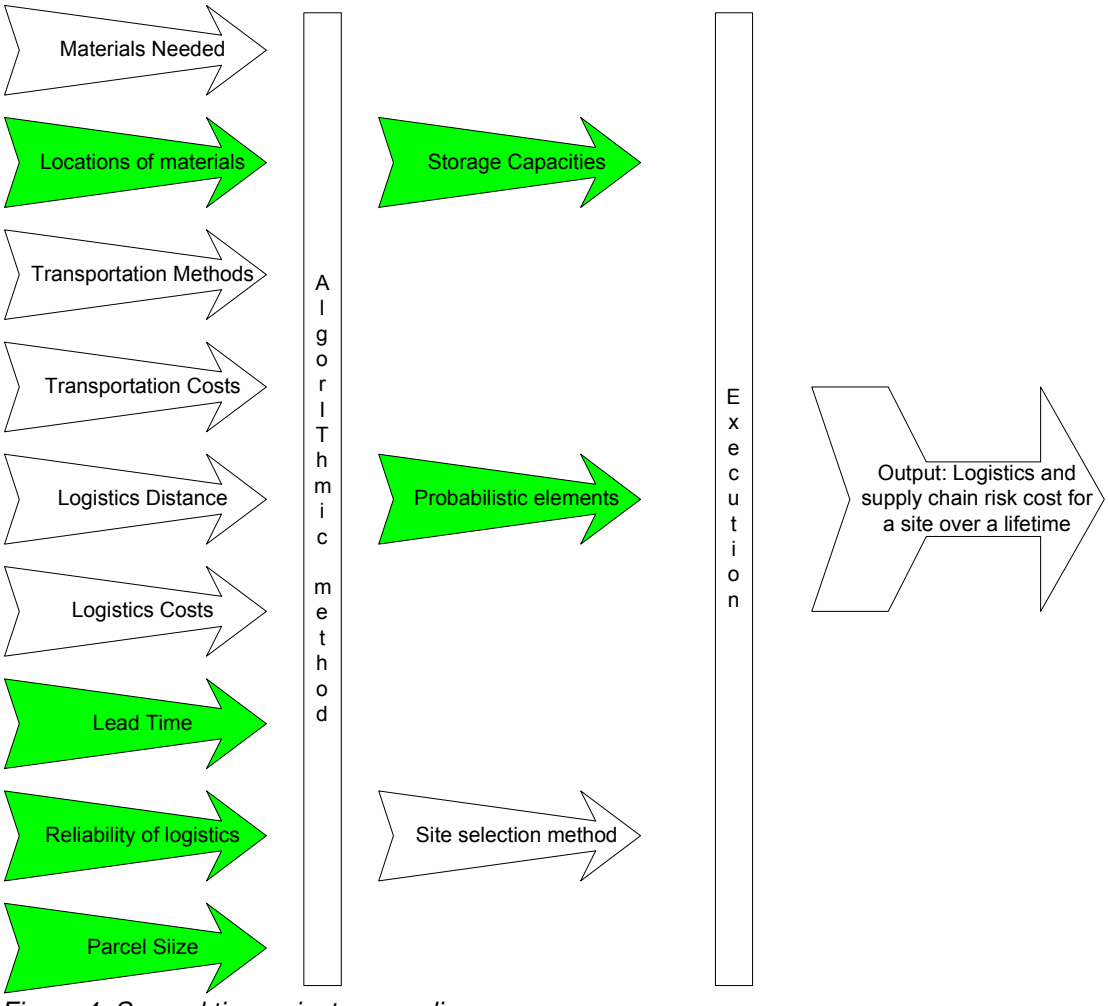


Figure 4: Second tier project scope diagram

3.3 Assumptions

- No other uncontrollable costs or logistics (such as waste disposal) that will be considered.
- All costs to be calculated in US$.
- Costs will be controlled using capex, opex and the NPV method using a 40 year operational lifetime.
- This project is done from a logistics chain perspective.

3.4 Solution Types [4]

3.4.1 A heuristic solution and exact solution

A distinction has to be made between a heuristic and exact solution.

A heuristic solution implies that the worst quality acceptable is used as the solution. It can be argued that the data sets from the real life problems are more inaccurate and a solution only has to be within a smaller confidence interval then that data. [12]

Exact solutions are globally optimum solutions within a set of criteria and are achievable within a discrete solution space.

3.4.2 A specific algorithm and general solver

A specific algorithm is designed to solve a problem specifically, while a general solver is an of-the-shelf product.

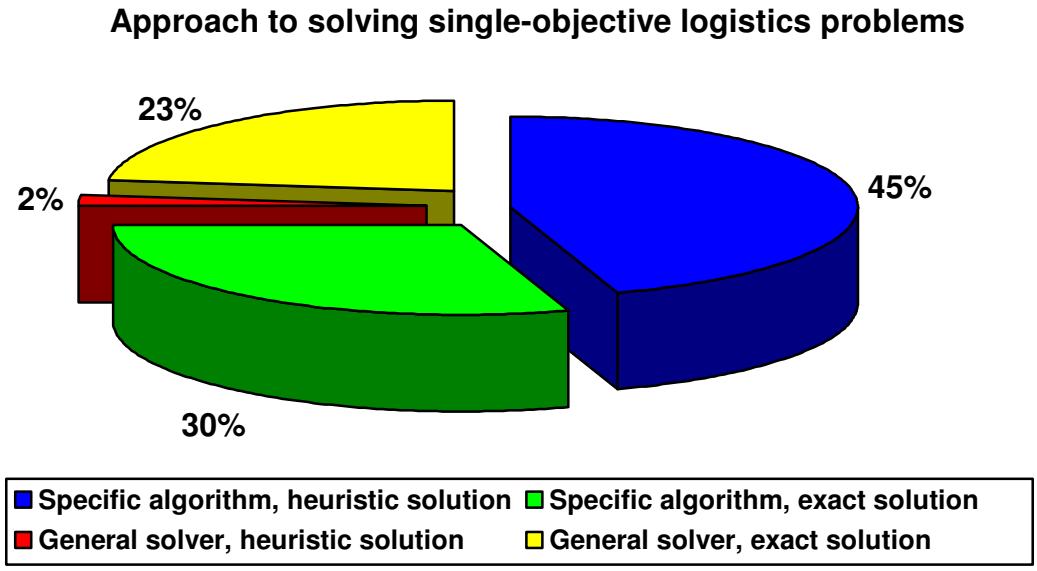


Figure 6: Approach to solving the problem

3.5 Application to problem

3.5.1 Solution type requirements

Features that are problem specific to this logistics study require an algorithm to solve. An exact solution can be had for tier one while tier two is more complex with multiple local optimums, so a heuristic solution will be used.

3.6 Supplementary methods and methodologies

3.6.1 The seven steps of the scientific method (Context)

The basis of this method is to solve a problem in a methodical manner. This method will be used to give structure to the process used. The schedule, this report and future reports will make reference to it.

These seven steps are:

1. identify the problem
2. make a hypothesis
3. create a solution
4. execute the solution
5. analyze data
6. review the solution
7. make a conclusion

This method will form the context in which the project will be completed.

3.6.2 Loose i2 programming method (Content)

This process was adapted from an informal process used by an i2 programmer at Sasol. The reason for the use of this process is to assure that the project gets executed in a lean and useful manner.

- Tier 1
  1. Unite business strategy with logistics chain strategy
  2. Identify measurement
  3. Get data
  4. Identify alternatives in terms of current criteria
     - Choose best alternative
- Tier 2
  5. Generate remaining feasible alternatives after new criteria has been incorporated
  6. Test different scenarios and infrastructure requirements
  7. Alternatives generation

The content of this project will have a correlation to this process.

3.6.3 Spiral model for reporting

This is the framework that will be used in report writing:

1. Report requirements will be identified by interviewing internal and external users.
2. A preliminary design is created to resolve possible misunderstandings and risks pertaining to the development.
3. A prototype will be constructed from the preliminary design as an approximation of the final report. After an evaluation is done, the report will be improved by backtracking to step one and two.
4. Final construction of the report.

4. Create a solution

4.1 Typical conceptual solution to current environment

4.1.1 Mathematical formulation example [9]

n = the number of facilities (departments).
v_{ij} = the number of unit loads moving between departments i and j.
u_{ij} = the cost to move a unit load a unit distance between departments i and j.
l_{ij} = the distance between departments i and j.
All u_{ij} elements of matrix U and all v_{ij} elements of matrix V are nonvarying

$$y_{ij} = u_{ij}v_{ij}$$

$$Y = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ & & \cdots & \\ y_{n1} & y_{n2} & \cdots & y_{nn} \end{bmatrix}$$

$$L = \begin{bmatrix} l_{11} & l_{12} & \cdots & l_{1n} \\ l_{21} & l_{22} & \cdots & l_{2n} \\ & & \cdots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix}$$

$$TC = \sum_{i=1}^{n} \sum_{j=1}^{n} y_{ij}l_{ij}$$

$$E = TC_0$$

$$x_{ik} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ if center } i \begin{pmatrix} \text{is} \\ \text{is not} \end{pmatrix} \text{ in location } k.$$

$$\begin{cases} \sum_i x_{ik} = 1 & \text{(all centers assigned)} \\ \sum_k x_{ik} = 1 & \text{(all locations occupied)} \\ x_{ik} = (0 \text{ or } 1) \end{cases}$$

$0 \leqq v_{i_1 i_2}$ = volume between center $i_1 \rightarrow$ center i_2

$0 \leqq l_{k_1 k_2}$ = distance from location $k_1 \rightarrow$ location k_2

$$E = \sum_{k_2} \sum_{k_1} \sum_{i_2} \sum_{i_1} x_{i_1 k_1} x_{i_2 k_2} y_{i_1 i_2} l_{k_1 k_2}$$

$$E_k = \sum_{i=1}^{n} \sum_{j=1}^{n} y_{ij}l_{ij}(K)$$

Figure 7: Conceptual Solution

4.2 New improved conceptual solutions

4.2.1 Mathematical formulation for tier one

y_{ij} = minimum cost over 40 years using the NPV to move an unit material, an unit distance between departments i and j

ue_{ijt} = the given cost to move an unit load of material between departments i and j with transporation method, t on existing infrastructure

un_{ijt} = the given cost to move an unit load of material between department i and j using transporation method, t on new infrastructure

v_{ij} = the given number of unit loads moving between department i and j per annum

l_{ijt} = the given distance between between departments i and j using logistics method, t

f_{ijt} = given $\begin{Bmatrix}1\\0\end{Bmatrix}$ if infrastructure $\begin{Bmatrix}\text{is}\\\text{is not}\end{Bmatrix}$ availible between departments i and j of logistics method t

fb_{ijt} = given $\begin{Bmatrix}1\\0\end{Bmatrix}$ if infrastructure $\begin{Bmatrix}\text{can}\\\text{can not}\end{Bmatrix}$ be built between departments i and j for transportation method t

fc_{ijt} = the given costant cost to build infrastructure between departments i and j using for logistics method t

fv_{ijt} = the given varible cost per unit distance to build infrastructure between departments i and j for logistics method t

c = the given NPV equaliser for an annual cost {33.33 for 40 years and ??20%?? per year}

A = Location Alternative for A = (1,2,..., Number of alternative sites)

TC_A = Total cost, for solution space A = (1,2,..., n) and n = number of feasible solutions

$$y_{ij} = \min\{f_{ijt} * [v_{ij} * ue_{ijt} * l_{ijt} * c] + fb_{ijt} * [(v_{ij} * un_{ijt} * l_{ijt} * c) + fc_{ijt} + fv_{ijt} * l_{ijt}]\} \text{ for all } i, j \text{ and } t \text{ in Solution Space } A \quad (1)$$

$$TC_A = \sum_{i=1}^{n}\sum_{j=1}^{n} y_{ij}(A) \text{ for all } A \quad (2)$$

$$TC_0 = \min(TC_A) \text{ for all } A \quad (3)$$

Figure 8: Improved conceptual solution

4.2.2 Notes

(1) Cost of moving the needed demand between departments
(2) Total logistics cost to place site a location A
(3) This is the minimum and most feasible solution cost

4.2.3 Improvements in new conceptual solution

- Multiple logistics modes and distance of travel for each
- Infrastructure availability incorporated
- Infrastructure and logistics costs in terms of capex, opex and the time value of money
- Arithmetic distinction between infrastructure that has to be built and that, which already has been built
- Exact solution
- Linear solution
- More intuitive

4.2.4 Mathematical formulation for tier two

$y_{ijt}(d)$ = minimum annual cost to move d materials with logistics method t,

where t = (road, rail, pipeline, conveyer, waterway) , between departments i and j

ue_{ijt} = the given opex cost to move an unit load material with logistics method t, on existing infrastructure

un_{ijt} = the given opex cost to move an unit load material with logistics method t, on new infrastructure

v_{ijmt} = the given number of unit loads moving on logistics method t per annuam

l_{ijt} = the given distance between departments for logistics method

f_{ijt} = given $\begin{Bmatrix}1\\0\end{Bmatrix}$ if infrastructure $\begin{Bmatrix}\text{is}\\\text{is not}\end{Bmatrix}$ availible between departments i and j, for logistics method t

p_{ijt} = given $\begin{Bmatrix}1\\0\end{Bmatrix}$ if infrastructure $\begin{Bmatrix}\text{can}\\\text{can not}\end{Bmatrix}$ be built between departments i and j, for logistics method t

fc_{ijt} = the given lump sum capex cost to build infrastructure for logistics method t

fv_{ijt} = the given varible capex per unit distance to build infrastructure for logistics method t

fe_{ijt} = the given lump sum capex cost to use existing infrastructure for logistics method t

c = the given NPV equaliser for an annual cost

e_n = the given single point failure cost for an n number of events, n = (1, 2, ..., number of failure events)

$e_{n'}'$ = the given multiple point failure saving given more the one points fail similtationsly

for n' number of cost saving oppertunities

p_n = the given annual probibility that a single point fails for an n number of events

i = the given department from where the material travels

j = the given department to where the material goes

Y_{ij} = the total cost to move all materials between given departmant i and j

Z_a = the total logistics chain cost between all departments for site a for a = (1,..., A)

A' = the chosen site

maB_t = the given maximum aamount of material that can move with travel method t.

miB_t = the given minimum amount of material that can move with travel method t.

Figure 9: Tier two variables for conception solution

$$y_{ijt}(d) = \min_{v_{ijmt}} \left\{ \sum_m c_{ijt}(v_{ijmt}) + \sum_m s_{ijt}(v_{ijmt}) + y_{ij(t+1)}(d - \sum_m g_{ijmt}[v_{ijmt}]) \right\}$$

$for\ \ all\ \ i, j\ and\ \ t \ldots(1)$

$rules\ \ :$

$miB_{ijt} < \sum_m v_{ijmt} < maB_{ijt}\ or\ \sum_m v_{ijmt} = 0 \ldots(2)$

$\sum_m v_{ijmt} = D \ldots(3)$

$y_{ij(waterway\ \ +1)}(d) = 0 \ldots(4)$

$p_{ijt} v_{ijmt} = v_{ijmt}\ \ for\ \ all\ \ m\ and\ \ all\ \ t \ldots(5)$

$0 <= \sum_m v_{ijmt} <= D \ldots(6)$

$$\sum_m c_{ijt}(v_{ijmt}) = \sum_m \{ p_{ijt} * f_{ijt} * [v_{ijmt} * l_{ijt} * c * ue_{ijt} + fe_{ijt} * c]$$

$$+ p_{ijt} * \overline{f_{ijt}} * [v_{ijmt} * l_{ijt} * c * un_{ijt} + fc_{ijt} + fv_{ijt} * l_{ijt}] \ldots(7)$$

$$\sum_t \sum_m s_{ijt}(v_{ijmt}) = e_n p_n - \sum_{n'} e_n'$$

+ inventory buffer cost + leadtime allowance + parchell size allowance. ..(8)

$Y_{ij} = \sum_t y_{ijt} \ldots(9)$

$Z_a = \sum_i \sum_j Y_{ij} \ldots(10)$

$Z_{A'} = \min\{ Z_a \}\ for\ \ all\ \ a \ldots(11)$

Figure 10: Tier two solution

4.2.5 Notes:

This method is known as deterministic dynamic programming and it makes use of the inverse knapsack problem to solve Z by minimising logistics costs for all possible logistics methods, materials and facility locations.
(1) The NPV cost to move materials for logistics method, between departments i and j
(2) The amount of materials moved with logistics method t needs to exceed the minimum units that can travel with that method, and be less the maximum amount allowed with that logistics method or it needs to be null.
(3)The amount of materials between departments needs to be exactly what the total demand for all logistics modes are between said departments.
(4)The programming at the last stage, after all logistics is done, cost must be null.
(5)All logistics modes used must be practical.
(6)The number of units within a single logistics method must be more then null and less then total demand.
(7) The logistics cost to move materials with logistics method t between departments i and j.
(8) The logistics chain risk cost between departments i and j.
(9) Total cost between departments i and j.
(10) Total logistics chain cost for solution space a.
(11) Best logistics chain solution.

4.2.6 Improvements over tier one solution

- The programming does not assume that all logistics modes are practical between all departments.
- All logistics modes have minimum and maximum bounds.
- The supply chain risk costs are also allocated.
- The lead-time, inventory and parcel-size extra costs are included.

5. Execution of solution

5.1 Basic information

5.1.1 Execution of solution summary

It became apparent early on that an information technology solution to locating a facility in a given space was needed, as the amount of data that had to be inputted and calculated with became too enormous to do by hand. A heuristic approach to inputting this data was created for the first tier and modified for the second tier. Calculations for the first tier were done using a simple algorithm that locally minimised the cost of the objective functions (each objective function is the logistics costs between two departments) for a specific set of input data. Calculations for the second tier were built around a deterministic dynamic programming approach to an inverse knapsack problem to calculate an objective function. A text file then serves as the result of calculations in a form of a report.

5.1.2 Programming Language

Excel was the first choice, as it is widely used and easy to operate. Drawbacks became evident as sizes of sets of information increased or decreased and programming became difficult. Sizes of matrixes are also limited to two dimensions, and input of a third or fourth dimension became user-unfriendly. Thus a different method was used.

Java was chosen as the solution execution method. Java evolved from C++, which evolved from C, which evolved BCPL and B. BCPL and B is a language that was written for the programming of operating systems and the first versions of UNIX was written in it. Today, C is the code used for general propose operating system programming. And C++, a spruced up version of C, had the capability of being object-orientated and is used for general purpose programming.

Java is revolutionary in that it proves reusable software components know as class objects together with a library of these objects. Being object-orientated, almost any English noun imaginable can be represented in both attributes (colours, name, and size) and behaviours (calculating, movement, and communication), making it popular, after the World Wide Web revolution of the nineteen-nineties.

Advantages of Java for in this project environment:

- Class library
- Flexible in calculation
- Flexible in input of complex sets of data
- Storage of data can be done in a structured manner
- Free to use

5.1.3 Programming philosophy

Emphasis in this report is the process that achieved the project outcomes. Little is said about the objects, variables, and other information technology mechanisms that are used in the approach.

The tier two scope program has the same structure as that of tier one. Extra input data is incorporated and execution of calculations is more comprehensive and sophisticated. (See mathematical formulations) The parts of the program that is only in the first tier scope will be highlighted in yellow and the parts that are only in the second tier scope will be highlighted in green.

The use case diagram is used to define to the reader of the report as to the options the user of the program has at each stage of the program execution.

The UML class diagram (UML: Unified Mark-up Language), a concise description of the classes, and the java code (diluted of intellectual property) is the only part of this report that will allude to or explain the information technology mechanisms that was used to achieve the outcomes.

5.2 Process of program execution:

5.2.1 Main execution

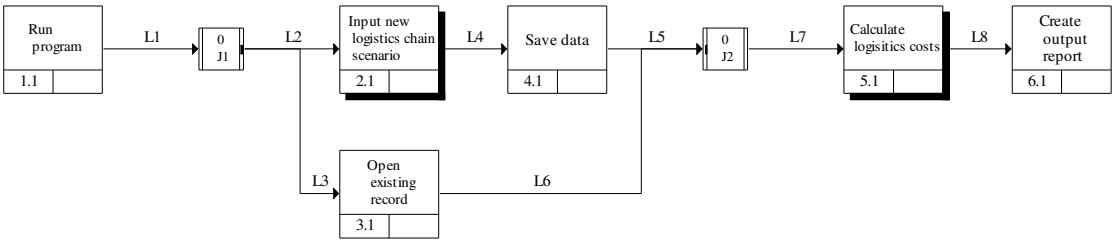


Figure 11: Main program execution diagram

5.2.2 Input data: 2.1

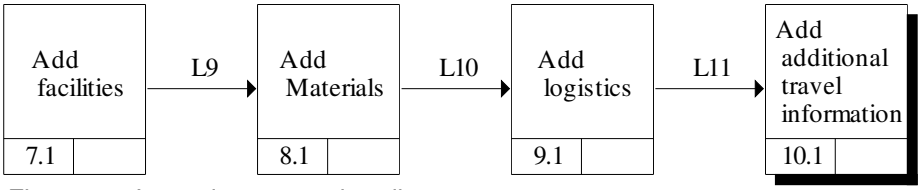


Figure 12: Input data execution diagram

5.2.3 Travel information: 10.1

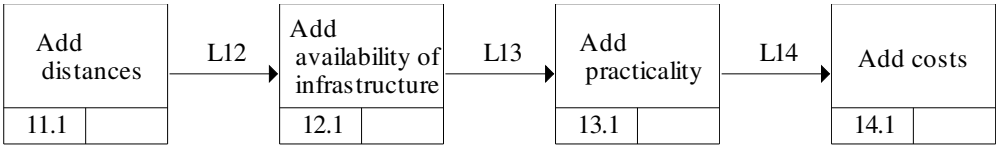


Figure 13: Additional travel information input execution diagram

5.2.4 Calculate logistics costs: 5.1

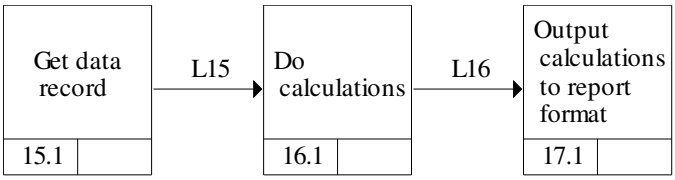
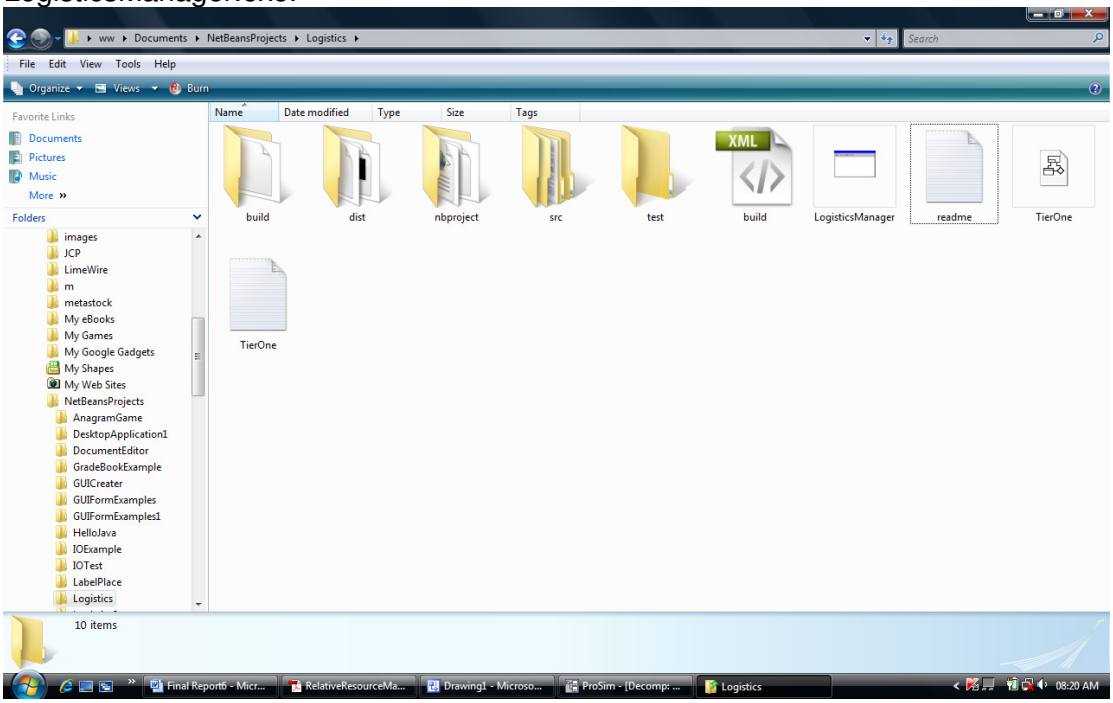


Figure 14: Calculations execution diagram

5.3 Program Description

5.3.1 Open Program

An executable file has been crated in the main program folder called LogisticsManager.exe.



Executing of the program is done by running this file. The opening screens for both tiers are shown below:

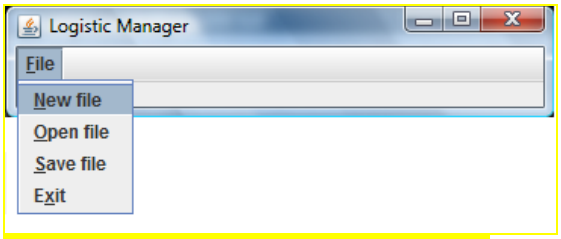


Figure15: Opening screen for the first tier

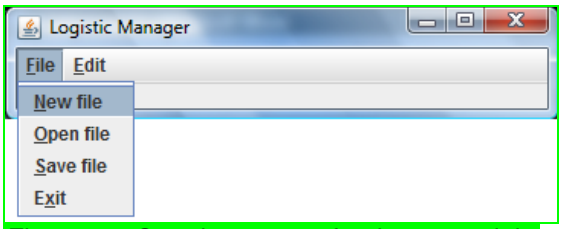


Figure 16: Opening screen for the second tier

From where one can create a new logistics chain scenario or open an existing record of a logistics chain by clicking “File”.

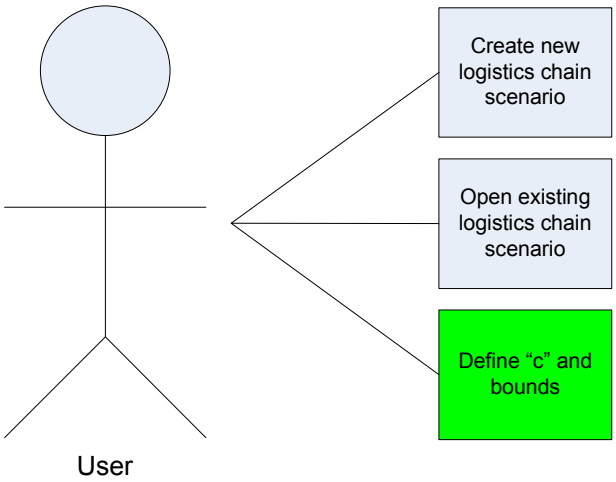
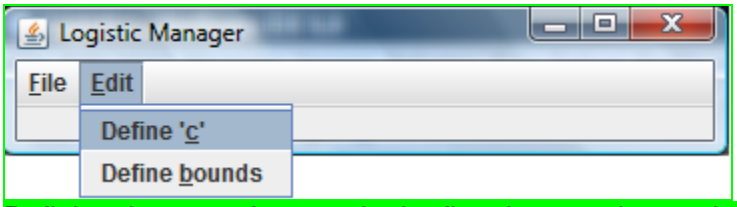


Figure16: Use case diagram when starting the program



Defining the cost of money in the first tier was done using a forty year time frame and a 20% annual decrease to the value of money. The second tier the user can edit the cost of money, by going to the edit menu and clicking on the *Define ‘c’* button.

Global maximum and minimum quantities that can be moved with a single logistics mode (road, rail, pipeline, conveyer, and waterway) can also be defined in the second tier by going to the edit menu and clicking on the *Define bounds* button.

5.3.2 Inputs

5.3.2.1 Inputs from existing file:

All inputs are saved in a *.ser file. This type of file is called a serialised record. After such a file has been saved, it can be opened again. The figure below shows the screen from where the user picks what logistics chain he wants to open. These files are in the main program folder.

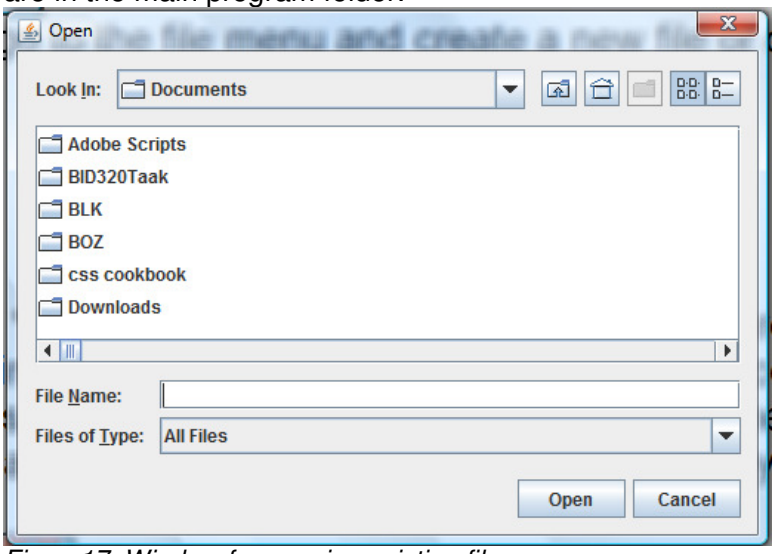


Figure17: Window for opening existing file

5.3.2.2 New data input:

The amount of data that has to be inputted is enormous. A heuristic has been created for inputting data in a user friendly manner.

Light blue operations in Travel info (2) are calculation done by the computer based on the criteria to only allow relevant data (only essential data) to be inputted by the user. Other decisions are done by the user:

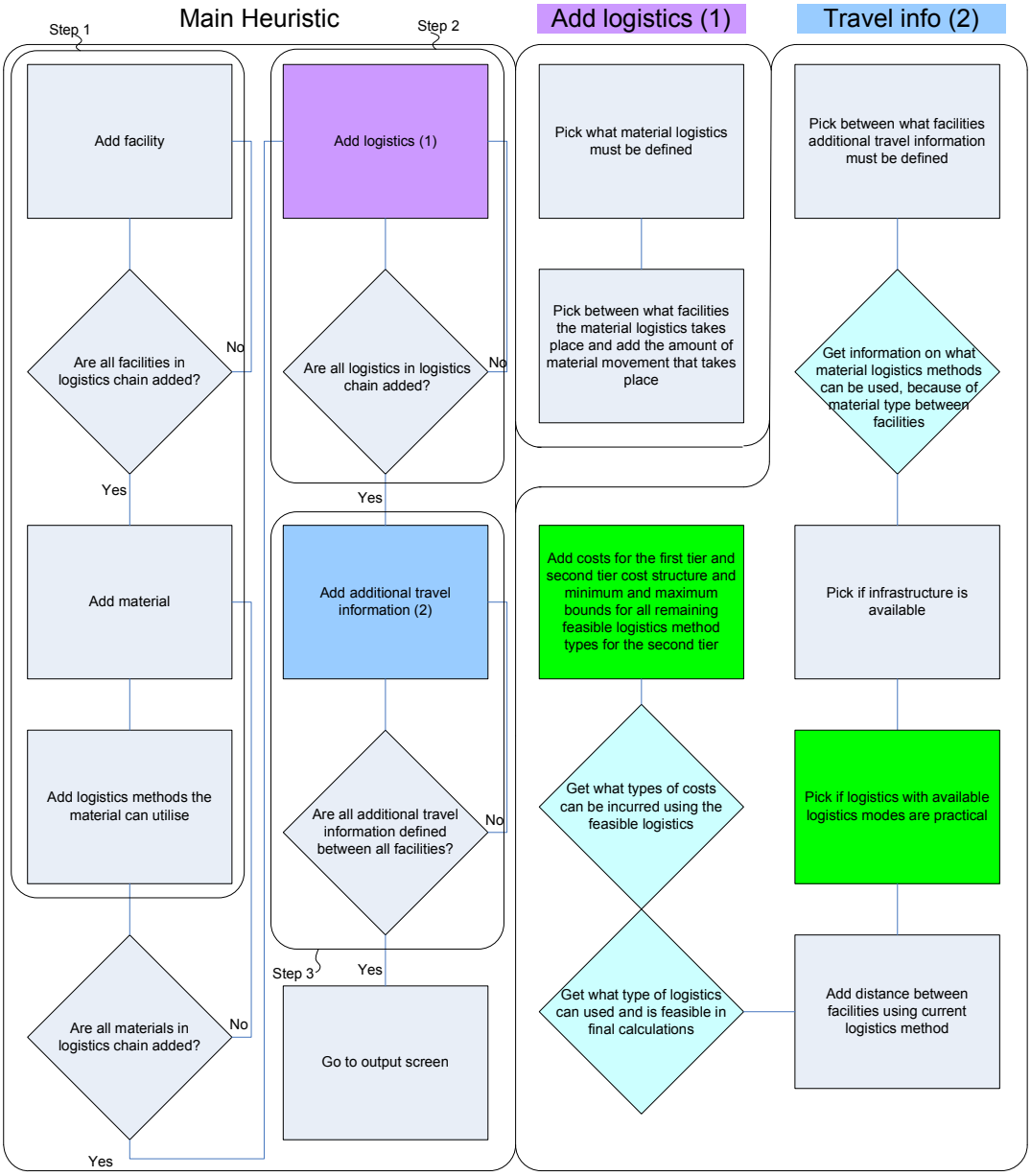


Figure 18: Input heuristic diagram

An input file is created as soon as a new file is opened. The first step (Step 1 in heuristic) is to define all facilities and material. It is done in the menu that appears called “Step 1 of 3”:

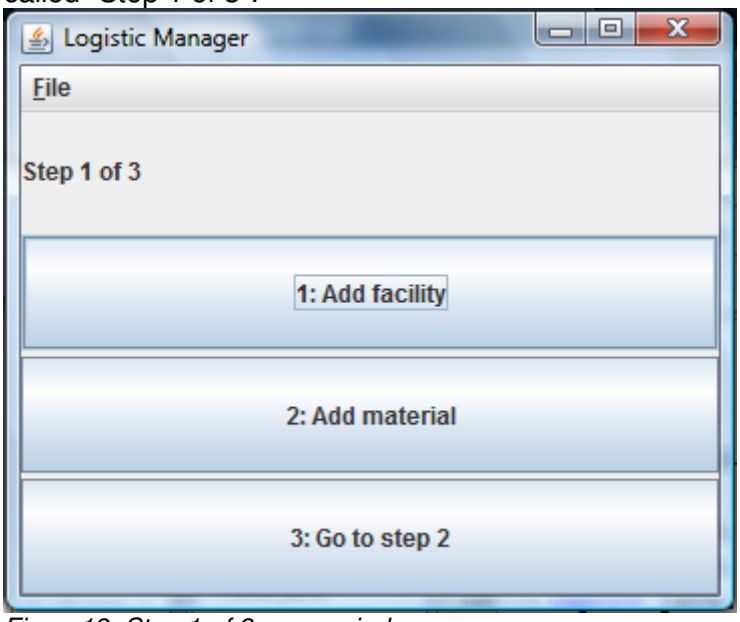


Figure19: Step 1 of 3 menu window

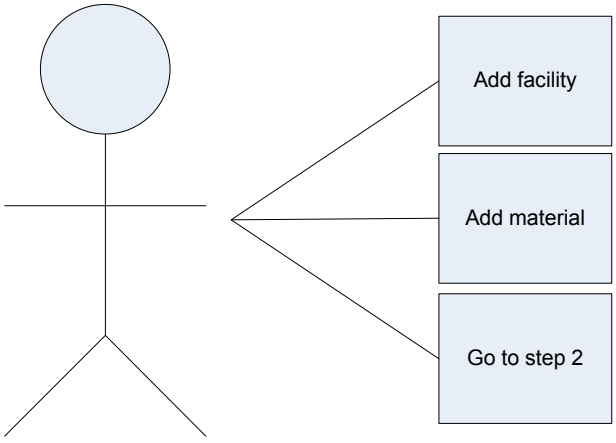


Figure20: Use case diagram for Step 1 of 3

Adding facilities: Clicking “1: Add facility”

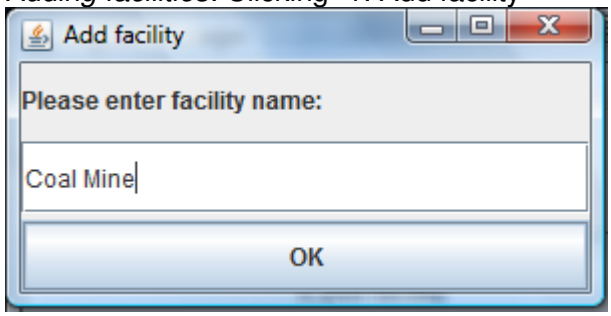


Figure21: Adding a facility window

For each new facility is added the user must fill in the new facility name in the “Add facility” menu. All facilities are defined now and in this way.

Adding materials: Clicking “2: Add material”:

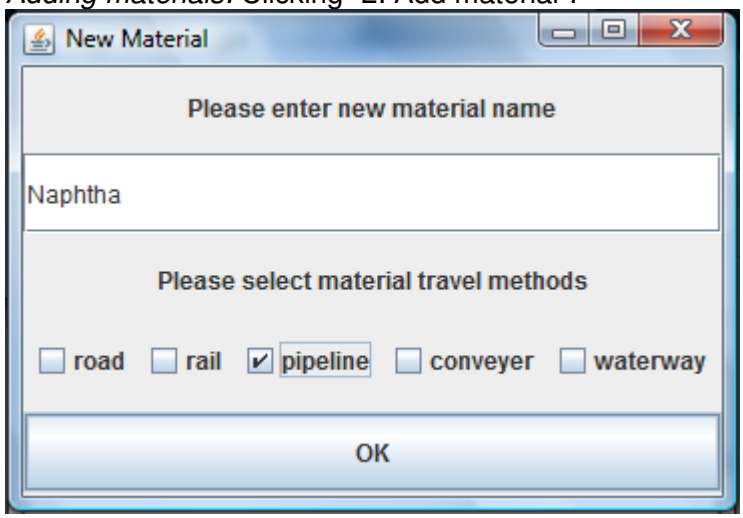


Figure22: New Material window

Enter the name of the material and which logistics method can be utilised to transport the material and click “OK”. For example; Naphtha can only be transported using a pipeline. All materials are defined now and in this way.

After materials and facilities have been defined, it is time to go to step 2 by clicking on the “3: Go to Step 2” button. A new menu will appear called the “Set logistics” or “Step 2 of 3” menu.

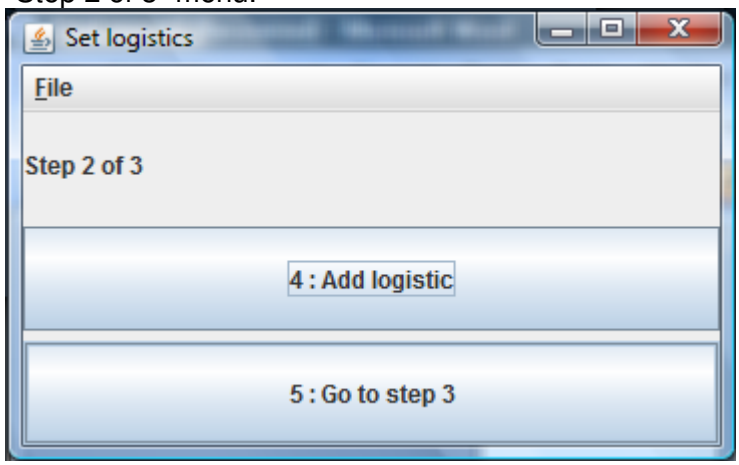


Figure23: Step 2 of 3

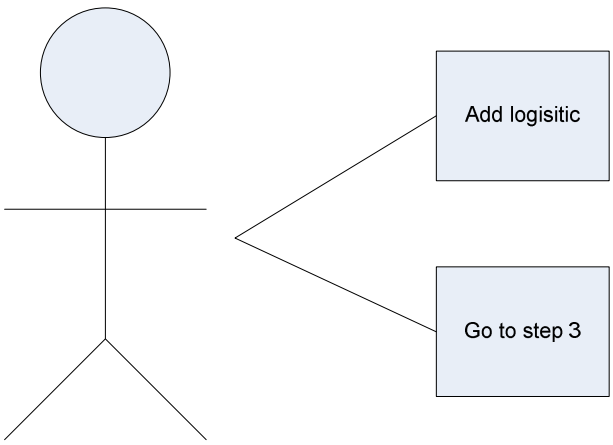
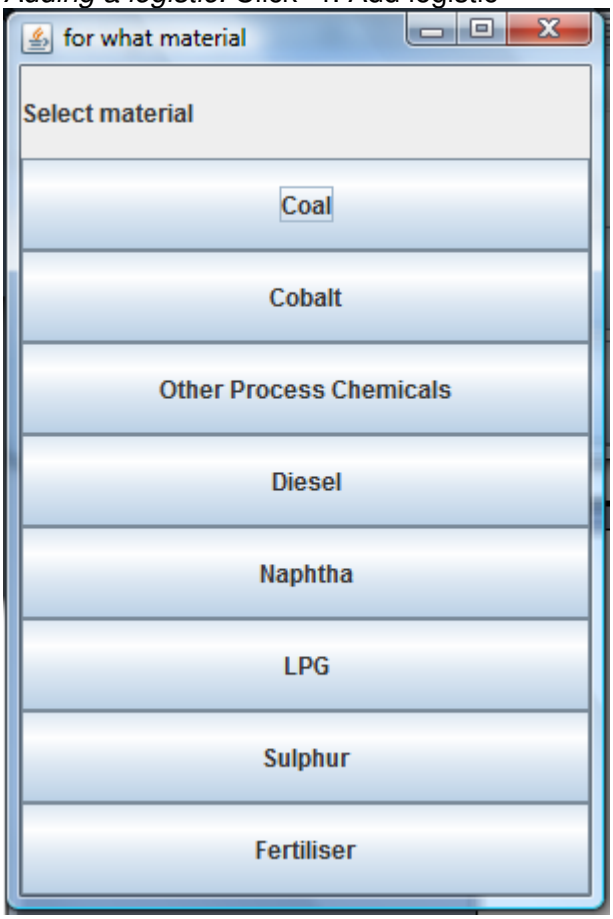


Figure24: Use case diagram for step 2

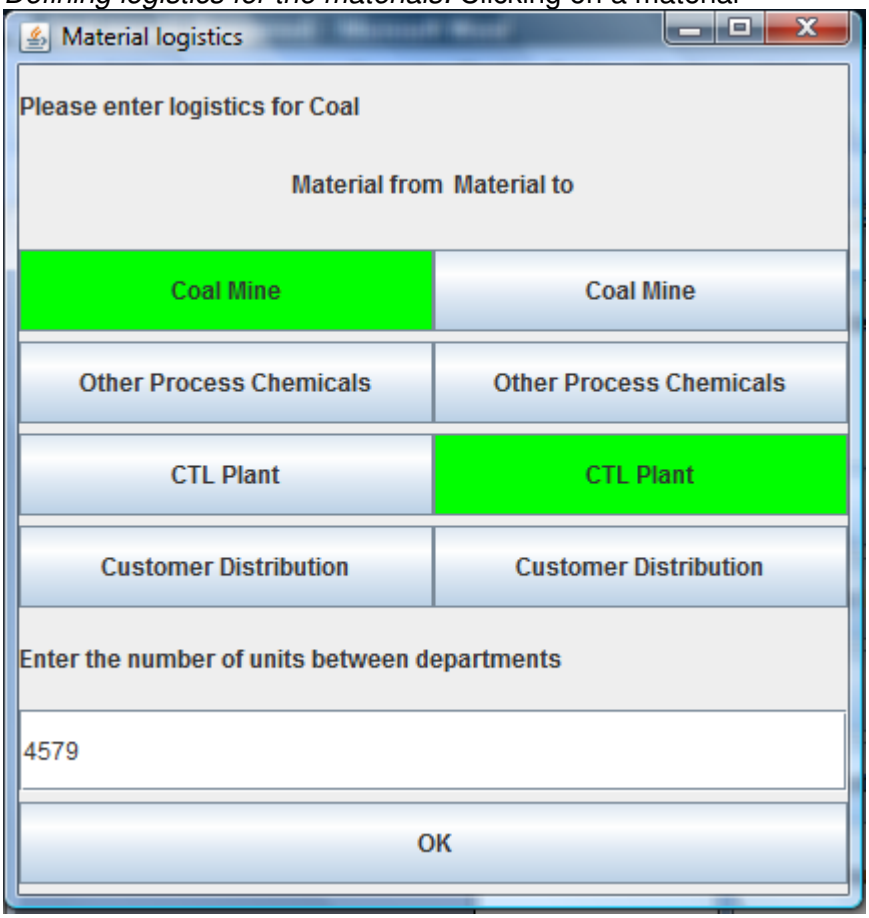
Material movement between facilities are inputted using step 2.

Adding a logistic: Click “4: Add logistic”



A menu appears from where the user selects what material logistics will be defined. Click on a material. All the material logistics in the logistics chain must to be defined in this way.

Defining logistics for the materials: Clicking on a material



Define material movement by selecting the facility from where the material are supplied from and the facility that demands the material, then entering the number of units of material moves between those facilities. If one material is transported between more then one pairs of facilities, the process is to be repeated. The unit type (for example: kg, m^3, tons, or litres) is at the user's discretion and is not defined in the program. Note only that one unit of material A and one unit of material B moving the same distance with the same logistics methods will result in the same cost.

All material movement is defined now, and in this way.

Add additional travel information: Clicking "5: Go to Step 3". A menu will appear called "Set distances".

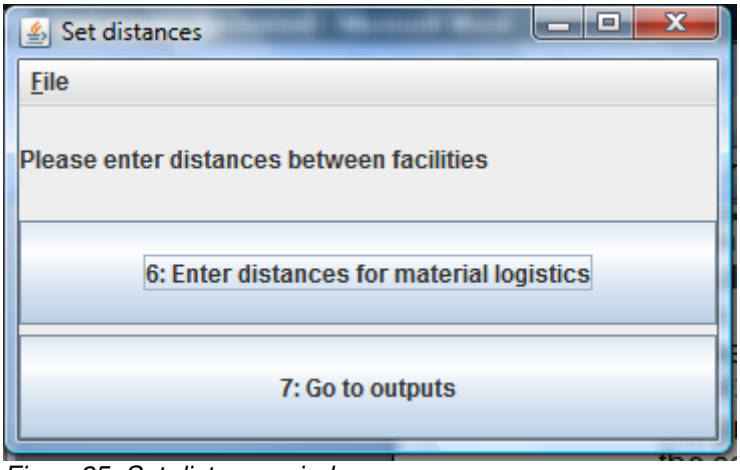


Figure25: Set distance window

In "Step 3 of 3" all *additional travel information* are defined. The process is repeated until all *additional travel information* has been inputted. The information includes:

- The distances between the facilities
- Infrastructure availability
- Practicality
- Travel costs

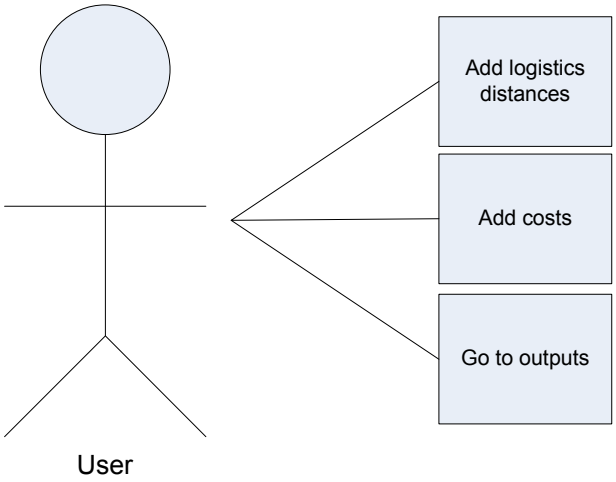
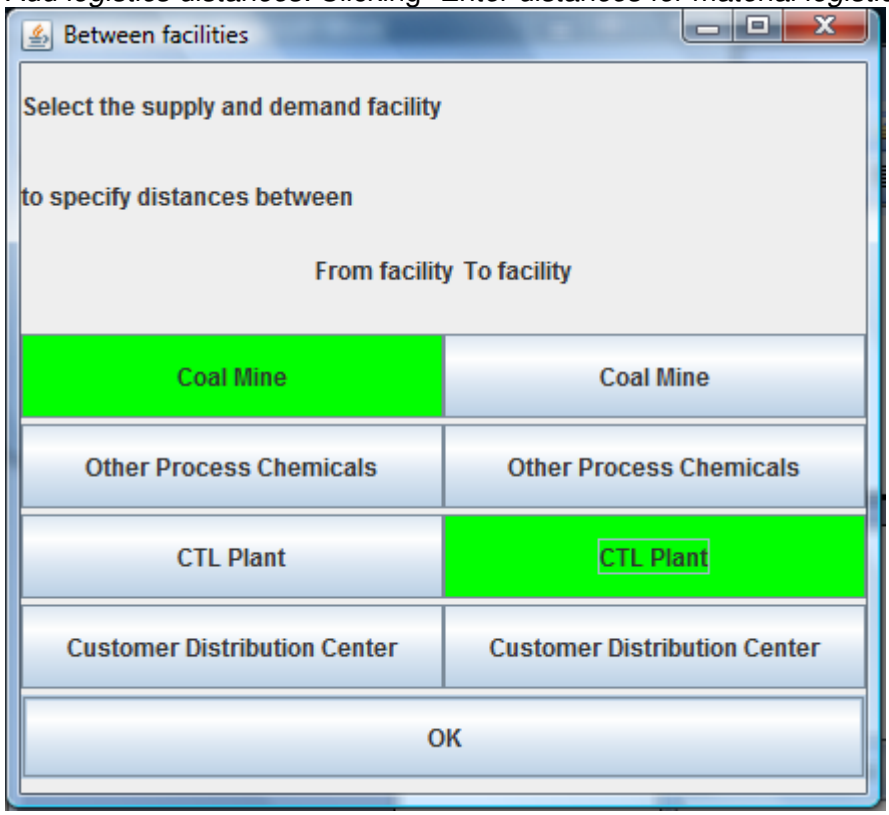


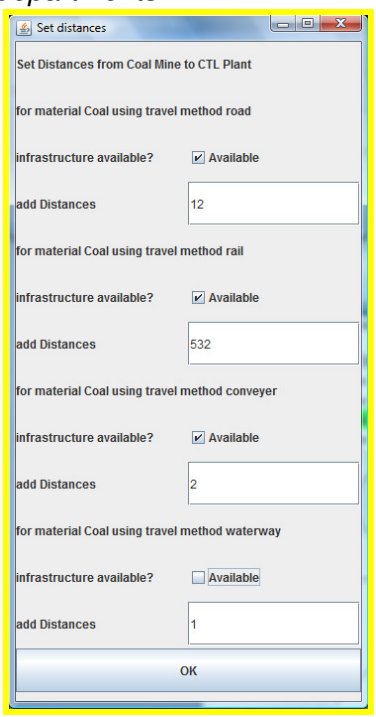
Figure26: Set distance window

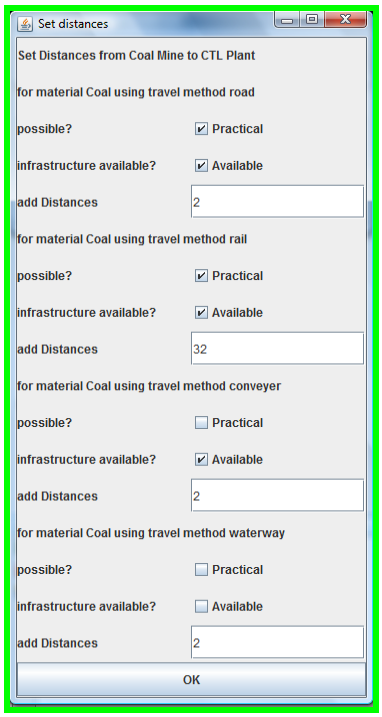
Add logistics distances: Clicking *"Enter distances for material logistics"*



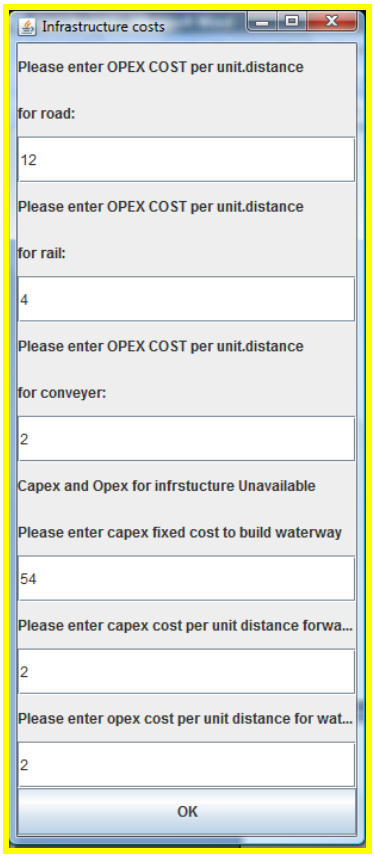
First select the facilities between which the additional travel information must be defined. Click on the "OK" button.

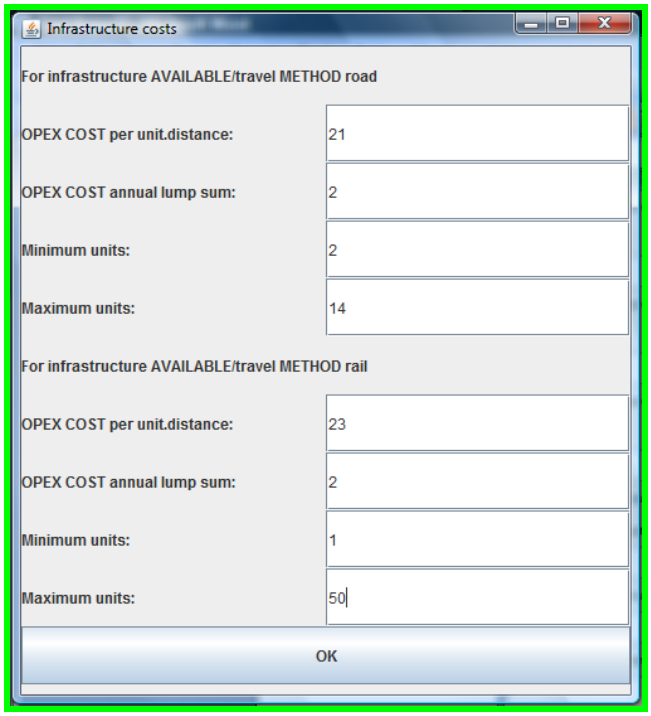
Input infrastructure availability, practicality, and the distances between selected departments





Add the distances required between facilities by clicking on the textbox (the white space) and entering the distance in integers. Also select whether infrastructure is available for current logistics method and if it is practical to move materials using this logistics method. After all distance information has been include click "OK".





Now costs must be included between the selected facilities. The second tier has an extra cost type called OPEX annual lump sum, together with the first tier OPEX unit.distance cost. The second tier also includes the minimum and maximum number of units that can be moved with the selected logistics method between selected facilities.

5.3.3 Storage of data

Data that has newly been added can now be saved in serialised record format. All the necessary logistics chain input data is stored in this single file record. The data can also later be reopened using the method described in *5.3.2.1 Inputs from existing file.*

Data is stored in the format and variables described by the mathematical model solutions for the first and second tier.

Data is stored using one of two methods:

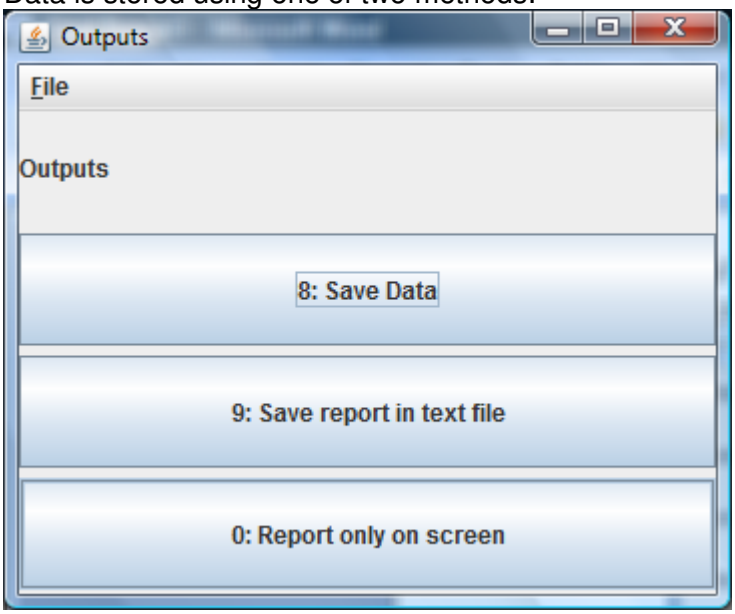


Figure27: Output menu

Method one: By clicking on the button “8: Save Data” on the “Outputs” menu.
Method two: By clicking on “File” then “Save Record”

Using both methods this window appears. The user enters the name of the file where the record is stored.

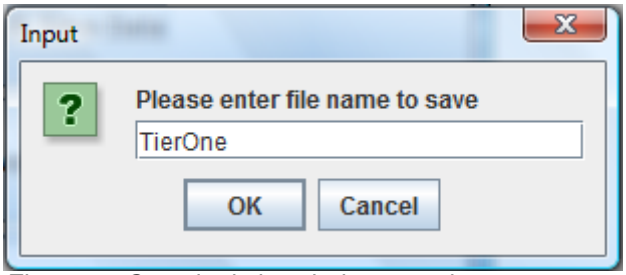


Figure28: Save logistics chain scenario

After a file name is entered, the file is created in the main program folder with a *.ser extension. A message then appears:

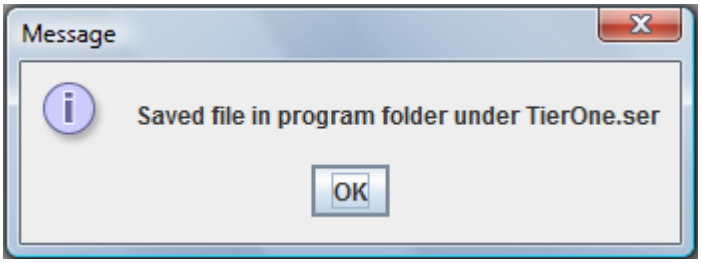
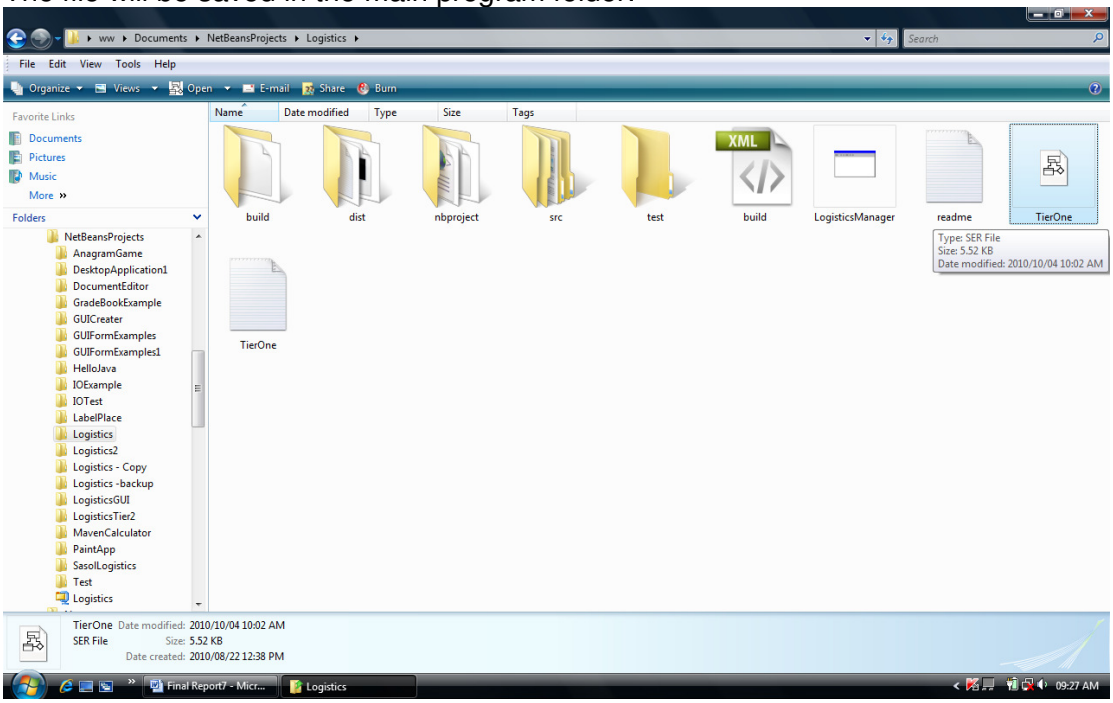


Figure29: Message after logistics chain scenario has been saved.

The file will be saved in the main program folder:



5.3.4 Outputs

5.3.3.1 Calculations

After a record has been created, calculations are done.

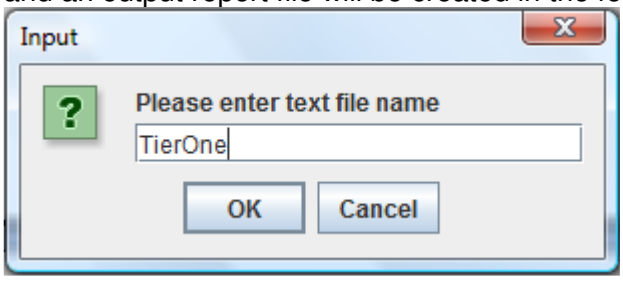
Tier one calculations: A simple algorithm is used to minimise logistics cost between all departments. After this is done the costs are added up and the total logistics chain cost are expressed in net present value terms.

Tier two calculations: Deterministic dynamic programming is used to solve an inverse (minimising the cost) knapsack problem, using Java, between all facilities. Additional information is also available and is incorporated.

See the mathematical solutions for the first and second tier, and the Java code class "Calculate" for execution method of the calculations.

5.3.3.2 Output report information

When clicking on the "9: Save report in text file" button, the calculations are done, and an output report file will be created in the form of a text file.



Information in output text file:

- Input information:
  - Materials are defined and the logistics methods that can be utilised
  - All facilities
  - Infrastructure availability between facilities for logistics methods
  - Number of units of material that moves between facilities
  - Logistic cost of moving materials
  - Minimum and maximum quantities of material that can be moved on each logistic method
  - Whether it is practical to use specific logistics methods between certain facilities
  - Logistics chain risks
- Output information for tier one:
  - Cost of moving materials between facilities for each logistics method
  - The minimum logistic cost between facilities
  - Total logistics cost of placing a CTL plant using the net present value method.

An example of a text files report (Information has been diluted of intellectual property):

```
MATERIAL: Coal USING METHOD: road rail conveyer waterway
MATERIAL: Cobalt USING METHOD: road rail waterway
MATERIAL: Other Process Chemicals USING METHOD: road rail waterway
MATERIAL: Diesel USING METHOD: pipeline
MATERIAL: Naphtha USING METHOD: pipeline
MATERIAL: LPG USING METHOD: pipeline
MATERIAL: Sulphur USING METHOD: road rail conveyer waterway
MATERIAL: Fertiliser USING METHOD: road rail conveyer waterway
FACILITIES: Coal Mine Other Process Chemicals CTL Plant Customer Distribution Center
INFRASTRUCTURE AVAILABILITY:
FOR TRAVEL METHOD road:
                            Coal Mine                   Other Process Chemicals     CTL Plant                   Customer
Distribution Center
Coal Mine                   false                       false                       true                        false
Other Process Chemicals     false                       false                       true                        false
CTL Plant                   false                       false                       false                       true
Customer Distribution Center false                      false                       false                       false
FOR TRAVEL METHOD rail:
                            Coal Mine                   Other Process Chemicals     CTL Plant                   Customer
Distribution Center
Coal Mine                   false                       false                       true                        false
Other Process Chemicals     false                       false                       true                        false
CTL Plant                   false                       false                       false                       true
Customer Distribution Center false                      false                       false                       false
FOR TRAVEL METHOD pipeline:
                            Coal Mine                   Other Process Chemicals     CTL Plant                   Customer
Distribution Center
Coal Mine                   false                       false                       false                       false
Other Process Chemicals     false                       false                       false                       false
CTL Plant                   false                       false                       false                       true
Customer Distribution Center false                      false                       false                       false
FOR TRAVEL METHOD conveyer:
                            Coal Mine                   Other Process Chemicals     CTL Plant                   Customer
Distribution Center
Coal Mine                   false                       false                       true                        false
Other Process Chemicals     false                       false                       false                       false
CTL Plant                   false                       false                       false                       true
Customer Distribution Center false                      false                       false                       false
FOR TRAVEL METHOD waterway:
                            Coal Mine                   Other Process Chemicals     CTL Plant                   Customer
Distribution Center
Coal Mine                   false                       false                       false                       false
Other Process Chemicals     false                       false                       true                        false
CTL Plant                   false                       false                       false                       true
Customer Distribution Center false                      false                       false                       false
UNITS BETWEEN FACILITIES:
FOR MATERIAL: Coal
                            Coal Mine                   Other Process Chemicals     CTL Plant                   Customer
Distribution Center
Coal Mine                   0                           0                           12                          0
Other Process Chemicals     0                           0                           0                           0
CTL Plant                   0                           0                           0                           0
Customer Distribution Center 0                          0                           0                           0
FOR MATERIAL: Cobalt
                            Coal Mine                   Other Process Chemicals     CTL Plant                   Customer
Distribution Center
Coal Mine                   0                           0                           0                           0
Other Process Chemicals     0                           0                           32                          0
CTL Plant                   0                           0                           0                           0
Customer Distribution Center 0                          0                           0                           0
FOR MATERIAL: Other Process Chemicals
                            Coal Mine                   Other Process Chemicals     CTL Plant                   Customer
Distribution Center
Coal Mine                   0                           0                           0                           0
Other Process Chemicals     0                           0                           169                         0
CTL Plant                   0                           0                           0                           0
Customer Distribution Center 0                          0                           0                           0
FOR MATERIAL: Diesel
                            Coal Mine                   Other Process Chemicals     CTL Plant                   Customer
Distribution Center
Coal Mine                   0                           0                           0                           0
Other Process Chemicals     0                           0                           0                           0
CTL Plant                   0                           0                           0                           543
Customer Distribution Center 0                          0                           0                           0
FOR MATERIAL: Naphtha
                            Coal Mine                   Other Process Chemicals     CTL Plant                   Customer
Distribution Center
Coal Mine                   0                           0                           0                           0
Other Process Chemicals     0                           0                           0                           0
CTL Plant                   0                           0                           0                           135
Customer Distribution Center 0                          0                           0                           0
FOR MATERIAL: LPG
                            Coal Mine                   Other Process Chemicals     CTL Plant                   Customer
Distribution Center
Coal Mine                   0                           0                           0                           0
Other Process Chemicals     0                           0                           0                           0
CTL Plant                   0                           0                           0                           85
Customer Distribution Center 0                          0                           0                           0
FOR MATERIAL: Sulphur
                            Coal Mine                   Other Process Chemicals     CTL Plant                   Customer
Distribution Center
Coal Mine                   0                           0                           0                           0
Other Process Chemicals     0                           0                           0                           0
CTL Plant                   0                           0                           0                           12
Customer Distribution Center 0                          0                           0                           0
FOR MATERIAL: Fertiliser
                            Coal Mine                   Other Process Chemicals     CTL Plant                   Customer
Distribution Center
Coal Mine                   0                           0                           0                           0
Other Process Chemicals     0                           0                           0                           0
CTL Plant                   0                           0                           0                           214
Customer Distribution Center 0                          0                           0                           0
road COST PER UNIT.DISTANCE ON AVAILABILE INFRASTRUCTURE:
                            Coal Mine                   Other Process Chemicals     CTL Plant                   Customer
Distribution Center
Coal Mine                   0                           0                           12                          0
Other Process Chemicals     0                           0                           1                           0
CTL Plant                   0                           0                           0                           1
Customer Distribution Center 0                          0                           0                           0
rail COST PER UNIT.DISTANCE ON AVAILABILE INFRASTRUCTURE:
                            Coal Mine                   Other Process Chemicals     CTL Plant                   Customer
Distribution Center
Coal Mine                   0                           0                           4                           0
Other Process Chemicals     0                           0                           1                           0
CTL Plant                   0                           0                           0                           2
Customer Distribution Center 0                          0                           0                           0
pipeline COST PER UNIT.DISTANCE ON AVAILABILE INFRASTRUCTURE:
                            Coal Mine                   Other Process Chemicals     CTL Plant                   Customer
Distribution Center
Coal Mine                   0                           0                           0                           0
```

```
Other Process Chemicals       0                         0                         0                         0
CTL Plant                     0                         0                         0                         2
Customer Distribution Center  0                         0                         0                         0
conveyer COST PER UNIT.DISTANCE ON AVAILABILE INFRASTRUCTURE:
                              Coal Mine                 Other Process Chemicals   CTL Plant                 Customer
Distribution Center
Coal Mine                     0                         0                         2                         0
Other Process Chemicals       0                         0                         0                         0
CTL Plant                     0                         0                         0                         3
Customer Distribution Center  0                         0                         0                         0
waterway COST PER UNIT.DISTANCE ON AVAILABILE INFRASTRUCTURE:
                              Coal Mine                 Other Process Chemicals   CTL Plant                 Customer
Distribution Center
Coal Mine                     0                         0                         0                         0
Other Process Chemicals       0                         0                         1                         0
CTL Plant                     0                         0                         0                         1
Customer Distribution Center  0                         0                         0                         0
waterway FIXED CAPEX COST ON UNAVAILABLE INFRASTRUCTURE:
                              Coal Mine                 Other Process Chemicals   CTL Plant                 Customer
Distribution Center
Coal Mine                     0                         0                         54                        0
Other Process Chemicals       0                         0                         0                         0
CTL Plant                     0                         0                         0                         0
Customer Distribution Center  0                         0                         0                         0
waterway CAPEX COST PER UNIT.DISTANCE ON UNAVAILABLE INFRASTRUCTURE:
                              Coal Mine                 Other Process Chemicals   CTL Plant                 Customer
Distribution Center
Coal Mine                     0                         0                         2                         0
Other Process Chemicals       0                         0                         0                         0
CTL Plant                     0                         0                         0                         0
Customer Distribution Center  0                         0                         0                         0
waterway OPEX COST PER UNIT.DISTANCE ON UNAVAILABLE INFRASTRUCTURE:
                              Coal Mine                 Other Process Chemicals   CTL Plant                 Customer
Distribution Center
Coal Mine                     0                         0                         2                         0
Other Process Chemicals       0                         0                         0                         0
CTL Plant                     0                         0                         0                         0
Customer Distribution Center  0                         0                         0                         0
DISTANCES BETWEEN FACILITIES:
FOR TRAVEL METHOD road:
                              Coal Mine                 Other Process Chemicals   CTL Plant                 Customer
Distribution Center
Coal Mine                     0                         0                         12                        0
Other Process Chemicals       0                         0                         12                        0
CTL Plant                     0                         0                         0                         1
Customer Distribution Center  0                         0                         0                         0
FOR TRAVEL METHOD rail:
                              Coal Mine                 Other Process Chemicals   CTL Plant                 Customer
Distribution Center
Coal Mine                     0                         0                         532                       0
Other Process Chemicals       0                         0                         1                         0
CTL Plant                     0                         0                         0                         2
Customer Distribution Center  0                         0                         0                         0
FOR TRAVEL METHOD pipeline:
                              Coal Mine                 Other Process Chemicals   CTL Plant                 Customer
Distribution Center
Coal Mine                     0                         0                         0                         0
Other Process Chemicals       0                         0                         0                         0
CTL Plant                     0                         0                         0                         12
Customer Distribution Center  0                         0                         0                         0
FOR TRAVEL METHOD conveyer:
                              Coal Mine                 Other Process Chemicals   CTL Plant                 Customer
Distribution Center
Coal Mine                     0                         0                         2                         0
Other Process Chemicals       0                         0                         0                         0
CTL Plant                     0                         0                         0                         1
Customer Distribution Center  0                         0                         0                         0
FOR TRAVEL METHOD waterway:
                              Coal Mine                 Other Process Chemicals   CTL Plant                 Customer
Distribution Center
Coal Mine                     0                         0                         1                         0
Other Process Chemicals       0                         0                         2                         0
CTL Plant                     0                         0                         0                         2
Customer Distribution Center  0                         0                         0                         0
for 'y':
57600.00: <from fac Coal Mine><to fac CTL Plant><material Coal><transport road>
851200.00: <from fac Coal Mine><to fac CTL Plant><material Coal><transport rail>
1600.00: <from fac Coal Mine><to fac CTL Plant><material Coal><transport conveyer>
856.00: <from fac Coal Mine><to fac CTL Plant><material Coal><transport waterway>
12800.00: <from fac Other Process Chemicals><to fac CTL Plant><material Cobalt><transport road>
1066.67: <from fac Other Process Chemicals><to fac CTL Plant><material Cobalt><transport rail>
2133.33: <from fac Other Process Chemicals><to fac CTL Plant><material Cobalt><transport waterway>
67600.00: <from fac Other Process Chemicals><to fac CTL Plant><material Other Process Chemicals><transport road>
5633.33: <from fac Other Process Chemicals><to fac CTL Plant><material Other Process Chemicals><transport rail>
11266.67: <from fac Other Process Chemicals><to fac CTL Plant><material Other Process Chemicals><transport waterway>
434400.00: <from fac CTL Plant><to fac Customer Distribution Center><material Diesel><transport pipeline>
108000.00: <from fac CTL Plant><to fac Customer Distribution Center><material Naphtha><transport pipeline>
68000.00: <from fac CTL Plant><to fac Customer Distribution Center><material LPG><transport pipeline>
400.00: <from fac CTL Plant><to fac Customer Distribution Center><material Sulphur><transport road>
1600.00: <from fac CTL Plant><to fac Customer Distribution Center><material Sulphur><transport rail>
1200.00: <from fac CTL Plant><to fac Customer Distribution Center><material Sulphur><transport conveyer>
800.00: <from fac CTL Plant><to fac Customer Distribution Center><material Sulphur><transport waterway>
7133.33: <from fac CTL Plant><to fac Customer Distribution Center><material Fertiliser><transport road>
28533.33: <from fac CTL Plant><to fac Customer Distribution Center><material Fertiliser><transport rail>
21400.00: <from fac CTL Plant><to fac Customer Distribution Center><material Fertiliser><transport conveyer>
14266.67: <from fac CTL Plant><to fac Customer Distribution Center><material Fertiliser><transport waterway>
for 'z':
856.00: <from fac Coal Mine><to fac CTL Plant><material Coal>
1066.67: <from fac Other Process Chemicals><to fac CTL Plant><material Cobalt>
5633.33: <from fac Other Process Chemicals><to fac CTL Plant><material Other Process Chemicals>
434400.00: <from fac CTL Plant><to fac Customer Distribution Center><material Diesel>
108000.00: <from fac CTL Plant><to fac Customer Distribution Center><material Naphtha>
68000.00: <from fac CTL Plant><to fac Customer Distribution Center><material LPG>
400.00: <from fac CTL Plant><to fac Customer Distribution Center><material Sulphur>
7133.33: <from fac CTL Plant><to fac Customer Distribution Center><material Fertiliser>
for 'Z':
856.00: <from fac Coal Mine><to fac CTL Plant>
6700.00: <from fac Other Process Chemicals><to fac CTL Plant>
617933.33: <from fac CTL Plant><to fac Customer Distribution Center>

NPV: 625489.33
```

Decisions that can be made from this report:

- The amount of material logistics on each logistics method between all facilities
- Minimum cost and logistics types of moving materials between facilities for all materials
- Comparing the total logistics chain scenarios costs to support the CTL plant placement decision

5.3.5 Limitations

5.3.4.1 Exception handling

No provision is made for exception handling. The data must be inputted correctly for the program to function. If done incorrectly, data is corrupted and the program crashes.

5.3.4.2 First time input

Each logistics chain scenario must be defined correctly the first time. The user must make sure that it is inputted correctly.

5.3.4.3 Saving data

Only complete sets of input data can be saved.

5.4 UML class diagrams, concise class description, Java code

Main Class

```
package logistics;

/*  Sasol and University of Pretoria
 *  Student Number 25245679
 *  Name Jakus van Rooyen
 */

public class logistics3  {
    public static void main(String[] args) {
        Menu menu = new Menu();
        menu.setMenu();

    }
}
```

Program Run Classes

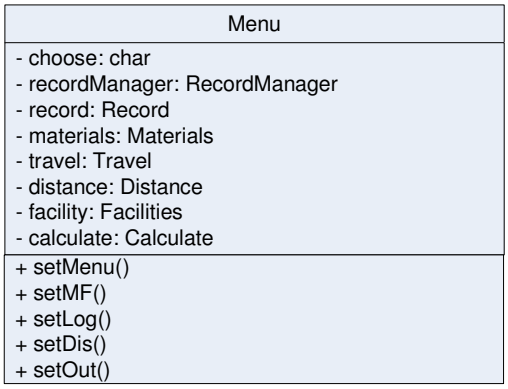


Figure31: Menu UML class diagram

Menu class

The menu class is the location where all other modes are launches. It contains the menus of the three input steps and the output menu. It also serves as temporary storage of most class objects before the data gets stored on the hard disc.

```
package logistics;

/*  Sasol and University of Pretoria
 *  Student Number 25245679
 *  Name Jakus van Rooyen
 */

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.SwingConstants;

public class Menu{

    private char choose;
    private JButton but[] = new JButton[10];
    private JLabel label[] = new JLabel[10];
    private JFrame jMenu, jOpen;
    private JMenu fileMenu;
    private JMenuItem newItem, openItem, saveItem, exitItem;
    private JMenuBar bar;

    private RecordManager recordManager = new RecordManager();
    private Record record = new Record();
    private Materials materials = new Materials();
    private Travel travel = new Travel();
    private TravelCosts travelCosts;
    private Distance distance = new Distance();
    private Facilities facility = new Facilities();
    private Calculate calculate= new Calculate();

    public Menu() {
    }

    public void setMenu () {

        recordManager = new RecordManager();
        record = new Record();

        jMenu = new JFrame( "Logistic Manager" );

        jMenu.setVisible( false );
        jMenu.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
```

```
        fileMenu = new JMenu( "File" );
        fileMenu.setMnemonic( 'F' );

        newItem = new JMenuItem( "New file" );
        newItem.setMnemonic( 'n' );
        fileMenu.add( newItem );
        newItem.addActionListener( new ActionListener() {
            public void actionPerformed( ActionEvent event ) {
                recordManager = new RecordManager();
                record = new Record();
                materials = new Materials();
                travel = new Travel();
                distance = new Distance();
                facility = new Facilities();
                calculate= new Calculate();
                recordManager.openNewFile();
                setMF();

            }
        });


        openItem = new JMenuItem( "Open file" );
        openItem.setMnemonic( 'o' );
        fileMenu.add( openItem );
        openItem.addActionListener( new ActionListener() {
            public void actionPerformed( ActionEvent event ) {
                recordManager = new RecordManager();
                record = recordManager.openExistingFile();
                setOut();

            }
        });

        saveItem = new JMenuItem( "Save file" );
        saveItem.setMnemonic( 's' );
        fileMenu.add( saveItem );
        saveItem.addActionListener( new ActionListener() {
            public void actionPerformed( ActionEvent event ) {
                recordManager = new RecordManager();
                record = new Record( facility.getFac(), materials, travel, distance );
                recordManager.saveFile( record );

            }
        });

        exitItem = new JMenuItem( "Exit" );
        exitItem.setMnemonic( 'x' );
        fileMenu.add( exitItem );
        exitItem.addActionListener( new ActionListener() {
            public void actionPerformed( ActionEvent event ) {
                System.exit( 0 );
            }
        });

        bar = new JMenuBar();
        jMenu.setJMenuBar( bar );
        bar.add( fileMenu );

        jMenu.setSize( 364, 75 );
        jMenu.setVisible( true );




    }

    public void setMF() {

        setMenu();
        jMenu.setVisible( false );
        jMenu.setLayout( new GridLayout( 4, 1,3 ,3 ) );
        jMenu.setSize( 364, 75*4 );
        ButtonHandler handler = new ButtonHandler();
        jMenu.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

        label[0] = new JLabel( "Step 1 of 3" );
        label[0].setAlignmentY( SwingConstants.CENTER );
        jMenu.add( label[0] );
        but[0] = new JButton( "1: Add facility" );
        jMenu.add( but[0] );
        but[0].addActionListener( handler );
        but[1] = new JButton( "2: Add material");
        jMenu.add( but[1] );
        but[1].addActionListener( handler );
        but[2] = new JButton( "3: Go to step 2");
        but[2].addActionListener( handler );
        jMenu.add( but[2] );

        bar = new JMenuBar();
        jMenu.setJMenuBar( bar );
        bar.add( fileMenu );

        jMenu.setVisible( true );
    }
    public void setLog() {

        ButtonHandler handler = new ButtonHandler();
        jMenu.setVisible( false );
        jMenu = new JFrame( "Set logistics" );
        jMenu.setSize( 364, 75*3 );
        jMenu.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        jMenu.setLayout( new GridLayout( 3, 1,5,5 ) );


        label[0] = new JLabel( "Step 2 of 3" );
        label[0].setAlignmentY( SwingConstants.CENTER );
        jMenu.add( label[0] );

        but[3] = new JButton( "4 : Add logistic" );
        but[3].addActionListener( handler );
        jMenu.add( but[3] );
        but[4] = new JButton( "5 : Go to step 3" );
        but[4].addActionListener( handler );
        jMenu.add( but[4] );

        bar = new JMenuBar();
        jMenu.setJMenuBar( bar );
        bar.add( fileMenu );

        jMenu.setVisible( true );
```

```
    }

    public void setDis() {
        ButtonHandler handler = new ButtonHandler();
        jMenu.setVisible( false );
        jMenu = new JFrame( "Set distances" );
        jMenu.setSize( 364, 75*3 );
        jMenu.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        jMenu.setLayout( new GridLayout( 3, 1,5,5 ) );
        jMenu.add( new JLabel( "Please enter distances between facilities" ) );
        but[5] = new JButton( "6: Enter distances for material logistics" );
        but[5].addActionListener( handler );
        jMenu.add( but[5] );
        but[6] = new JButton( "7: Go to outputs");
        but[6].addActionListener( handler );
        jMenu.add( but[6] );
        bar = new JMenuBar();
        jMenu.setJMenuBar( bar );
        bar.add( fileMenu );
        jMenu.setVisible( true );
    }

    public void setOut() {
        jMenu.setVisible( false );
        jMenu = new JFrame( "Outputs" );
        jMenu.setSize( 364, 75*4 );
        jMenu.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        jMenu.setLayout( new GridLayout( 4, 1,5,5 ) );
        jMenu.add( new JLabel( "Outputs" ) );
        but[7] = new JButton( "8: Save Data" );
        ButtonHandler handler = new ButtonHandler();
        but[7].addActionListener( handler );
        jMenu.add( but[7] );
        but[8]= new JButton( "9: Save report in text file" );
        but[8].addActionListener( handler );
        jMenu.add( but[8] );
        but[9] = new JButton( "0: Report only on screen" );
        but[9].addActionListener( handler );
        jMenu.add( but[9] );
        bar = new JMenuBar();
        jMenu.setJMenuBar( bar );
        bar.add( fileMenu );

        jMenu.setVisible( true );

    }



    private class ButtonHandler implements ActionListener {
        public void actionPerformed ( ActionEvent event ) {
            choose = event.getActionCommand().charAt(0);
            switch ( choose ) {
                case '1' :
                    facility.setFacility();
                    break;
                case '2' :
                    materials.setMaterial();
                    break;
                case '3':
                    setLog();
                    break;
                case '4':
                    travel.setTravel( facility.getFac(), materials.getMat() );

                    break;
                case '5':
                    setDis();
                    break;
                case '6':
                    distance.setD( facility.getFac(), materials, travel );
                    break;
                case '7':
                    setOut();
                    System.out.println( "Marker 1" );

                    record = new Record( facility.getFac(), materials, travel, distance );
                    for ( int f1 = 0; f1 < facility.getFac().size(); f1++ ) {
                        for ( int f2 = 0; f2 <  facility.getFac().size(); f2++ ) {
                            for ( int t = 0; t < 5; t++ ) {
                                System.out.printf( "ue[%d][%d][%d] = %d\n", f1, f2, t, record.getUE( f1, f2, t ) );
                            }
                        }
                    }
                    break;
                case '8':
                    recordManager = new RecordManager();
                    record = new Record( facility.getFac(), materials, travel, distance );
                    recordManager.saveFile( record );
                    System.out.println( "Marker 2");
                    break;
                case '9':
                    recordManager.closeFile();
                    calculate.calNPV( record );
                    calculate.printTextRecord( record );
                    recordManager.closeFile();

                    break;
                case '0' :
                    recordManager.closeFile();
                    calculate.calNPV( record );
                    calculate.printGUIRecord( record );
                    recordManager.closeFile();
                    break;


            }
        }
    }
}
```

Input Classes

Facilities
- getF: String - fac: ArrayList<String>
+ setFacility() + getFac(): ArrayList<String>

Figure32: Facilities UML class diagram

Facilities class

This is where the input of facilities occurs.

```
package logistics;

/*  Sasol and University of Pretoria
 *  Student Number 25245679
 *  Name Jakus van Rooyen
 */

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;

/**
 *
 * @author jakus
 */
public class Facilities {
    private String getF;
    private JFrame jFac;
    private JLabel jLabel;
    private JTextField jTextField;
    private JButton jButton;

    private List<String> fac = new ArrayList<String>();

    public void setFacility() {
        jFac = new JFrame( "Add facility" );
        jLabel = new JLabel( "Please enter facility name: " );
        jFac.add( jLabel );
        jTextField = new JTextField( "" );
        jFac.add( jTextField );
        jButton = new JButton ( "OK" );
        jButton.addActionListener( new ActionListener() {
            public void actionPerformed( ActionEvent event ) {
                getF = jTextField.getText();
                fac.add( getF );
                jFac.setVisible( false );
            }
        });

        jFac.add( jButton );

        jFac.setSize( 300, 50 * 3 );
        jFac.setLayout( new GridLayout( 3, 1, 3, 3 ) );
        jFac.setDefaultCloseOperation( JFrame.HIDE_ON_CLOSE );
        jFac.setVisible( true );


    }

    public List <String> getFac() {
        return fac;
    }
}
```

Materials class

Materials
- getM: String - tTravel[ ] : String = {“road”,”rail”,”pipeline”,”conveyer”,”waterway”} - mt[ ][ ]: boolean - place[ ][ ]: boolean - b: boolean - mat1: ArrayList <String>
+ setMaterials() + getMaterials() + setMT() + getMT() + getMTa() + getMat()

Figure33: Materials UML class diagram

All the materials, and logistics modes that can be utilised by the materials, are inputted using this class.

```
package logistics;

/*  Sasol and University of Pretoria
 *  Student Number 25245679
 *  Name Jakus van Rooyen
 */

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SwingConstants;

public class Materials {
    private String getM;
    private boolean mt[][], place[][], b;
    private String tTravel[] = { "road", "rail", "pipeline", "conveyer", "waterway" };

    private JPanel jPanel;

    private JTextField JGetM;
    private JCheckBox travel[];
    private JButton OKButton;
    private JLabel mes1;
    private JFrame jMat;

    private List <String> mat1 = new ArrayList <String>();


    public void setMaterial() {
        jMat = new JFrame( "New Material" );
        jMat.setLayout( new GridLayout( 5, 1, 3 ,3 ) );
        jMat.setVisible( true );
        jMat.setDefaultCloseOperation( JFrame.HIDE_ON_CLOSE );
        jMat.setSize( 365, 5*50 );
        mes1 = new JLabel( "Please enter new material name" );
        mes1.setHorizontalAlignment( SwingConstants.CENTER );
        jMat.add( mes1 );
        JGetM = new JTextField( "" );
        jMat.add( JGetM );
        mes1 = new JLabel( "Please select material travel methods" );
        mes1.setHorizontalAlignment( SwingConstants.CENTER );
        jMat.add( mes1 );
        travel = new JCheckBox[5];
        jPanel = new JPanel();
        jPanel.setLayout( new FlowLayout() );
        for ( int t = 0; t < 5; t++ ) {
            travel[t] = new JCheckBox( tTravel[t] );
            jPanel.add( travel[t] );
        }
        jMat.add( jPanel );
        OKButton = new JButton( "OK" );
        Handler handler = new Handler();
        jMat.add( OKButton );
        OKButton.addActionListener( handler );
    }
    public String getMaterial() {
        return getM;
    }
    private void setMT() {

        if ( b == false ) {
            mt = new boolean[1][5];
        }
        else {
            place = new boolean[mt.length+1][5];
            for ( int counter1 = 0; counter1 < mt.length; counter1++ ) {
                for ( int counter2 = 0; counter2 < 5; counter2++ ) {
                    place[counter1][counter2] = mt[counter1][counter2];
                }
            }
            mt = new boolean[place.length][5];
            mt = place;
        }
        b = true;
```

```
    }
    public boolean getMT ( int  i1, int i2 ) {
        return mt[i1][i2];
    }
    public boolean[][] getMTa() {
        return mt;
    }

    public List <String> getMat() {
        return mat1;
    }
    public class Handler implements ActionListener {
        public void actionPerformed ( ActionEvent event ) {
            getM = JGetM.getText();
            JGetM = new JTextField("");
            mat1.add( getM );
            setMT();
            for ( int t = 0; t < 5; t++ ) {
                if ( travel[t].isSelected() ) {
                    mt[mt.length-1][t] = true;
                }
                else {
                    mt[mt.length-1][t] = false;
                }
            }
            jMat.setVisible( false );
        }
    }
}
```

Travel class

Travel
- i: integer - j: integer - m: integer - units[ ][ ][ ]: integer - place[ ][ ][ ]: integer - fac: ArrayList <String> - b: boolean - mat: ArrayList <String> - string: String = “”
+ setTravel(fac: ArrayList <String>, mat: ArrayLis <String>) + setMatMenu(mat: ArrayList <String>) + defineV(fac: ArrayList <String>, mat: ArrayList <String>) + getV(i1: integer, i2: integer, i3: integer): integer + getVa() : integer[ ][ ][ ]

Figure34: Travel UML class diagram

The amount of materials that moves between departments is inputted using in this class.

```
package logistics;

/*  Sasol and University of Pretoria
 *  Student Number 25245679
 *  Name Jakus van Rooyen
 */

import java.awt.Color;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SwingConstants;

public class Travel {
    private int i, j, k, m, units[][][], place[][][], counter = 0;
    private JPanel jPanel, jPanelCom ;
    private JLabel jLabel, jLabel2, jLabel3;
    private JButton bFac[][], bMat[], OK;
    private List<String> fac1;
    private JTextField unitsText;
    private boolean b;
    private JFrame jTravel;
    private List<String> mat1;

    private String string = "";

    public void setTravel( List< String > fac, List< String > mat ) {
        jTravel = new JFrame( "Material logistics" );
        defineV( fac, mat );
        jTravel.setLayout( new GridLayout( fac.size() + 5 ,1,5,5 ) );
        jTravel.setVisible( false );
        jTravel.setDefaultCloseOperation( JFrame.HIDE_ON_CLOSE );
        jTravel.setSize( 300, ( fac.size()+5 ) *50 );
        fac1 = fac;
        setMatMenu( mat );
        jLabel = new JLabel( "Please enter logistics for " + mat.get( m ) );
        jTravel.add( jLabel );
        jLabel = new JLabel( "Material from");
        jLabel.setAlignmentY( SwingConstants.CENTER );
        jLabel2 = new JLabel( "Material to" );
        jLabel2.setAlignmentY( SwingConstants.CENTER );
```

```
        jPanel = new JPanel();
        jPanel.add( jLabel );
        jPanel.add( jLabel2 );
        jTravel.add( jPanel);
        bFac = new JButton[fac.size()][2];
        ButtonHandlerTo handlerTo = new ButtonHandlerTo();
        ButtonHandlerFrom handlerFrom = new ButtonHandlerFrom();
        for ( int i1 = 0; i1 < fac.size(); i1++ ) {
            jPanel = new JPanel();
            jPanel.setLayout( new GridLayout( 1, 2) );
            bFac[i1][0] = new JButton( fac.get( i1 ) );
            bFac[i1][1] = new JButton( fac.get( i1 ) );
            bFac[i1][0].addActionListener( handlerFrom );
            bFac[i1][1].addActionListener( handlerTo );
            jPanel.add( bFac[i1][0] );
            jPanel.add( bFac[i1][1] );
            jTravel.add( jPanel );
        }
        jLabel = new JLabel( "Enter the number of units between departments" );
        jTravel.add( jLabel );

        unitsText= new JTextField();

        OK = new JButton ( "OK" );
        OKHandler oKHandler = new OKHandler();
        OK.addActionListener( oKHandler );
        jTravel.add( unitsText );
        jTravel.add( OK );
    }

    private JFrame matMenu;

    private void setMatMenu ( List<String> mat ) {
        matMenu = new JFrame( "for what material");
        matMenu.setDefaultCloseOperation( JFrame.HIDE_ON_CLOSE );
        matMenu.setSize( 300, 50 + mat.size()*50 );
        matMenu.setLayout( new GridLayout( mat.size() +1, 1 ) );
        matMenu.setVisible( true );

        jLabel = new JLabel( "Select material" );

        matMenu.add( jLabel );

        bMat = new JButton[mat.size()];
        string = "";

        for ( int m1 = 0; m1 < mat.size(); m1++ ) {
            bMat[m1] = new JButton( mat.get( m1 ) );
            matMenu.add( bMat[m1] );
            mat1 = mat;
            bMat[m1].addActionListener( new ActionListener() {
                public void actionPerformed( ActionEvent event ) {
                    string = event.getActionCommand();
                    for ( int m2 = 0; m2 < mat1.size(); m2++ ) {
                        if ( string.equals(mat1.get(m2)) ) {
                            m = m2;
                        }
                    }
                    matMenu.setVisible( false );
                    jTravel.setVisible( true );
                }
            });
        }

    }




    private void defineV( List<String> fac, List<String> mat ) {

        i = -1;
        j = -1;
        if ( !b ) {
            units = new int[fac.size()][fac.size()][mat.size()];
            place = new int[fac.size()][fac.size()][mat.size()];
        }
        else {
            place = new int[units.length][units.length][];
            for ( int counter1 = 0; counter1 < units.length; counter1++ ) {
                for ( int counter2 = 0; counter2 < units.length; counter2++ ) {
                    place[counter1][counter2] = new int[units[counter1][counter2].length];
                }
            }
            for ( int counter1 = 0; counter1 < place.length; counter1++ )
            for ( int counter2 = 0; counter2 < place.length; counter2++ )
            for ( int counter3 = 0; counter3 < place[counter1][counter2].length; counter3++ ) {
                place[counter1][counter2][counter3] = units[counter1][counter2][counter3];
            }
        }
        units = new int[fac.size()][fac.size()][mat.size()];

        for ( int counter1 = 0; counter1 < place.length; counter1++ ) {
            for ( int counter2 = 0; counter2 < place.length; counter2++ ) {
                for ( int counter3 = 0; counter3 < place[counter1][counter2].length; counter3++ ) {
                    if ( place[counter1][counter2][counter3] > 0 )
                        units[counter1][counter2][counter3] = place[counter1][counter2][counter3];
                    else {
                        units[counter1][counter2][counter3] = 0;
                    }
                }
            }
        }
        b = true;
    }

    public int getV( int i1, int i2, int i3 ) {
        return units[i1][i2][i3];
    }

    public int[][][] getVa() {
        return units;
    }



    private class ButtonHandlerFrom implements ActionListener {
        public void actionPerformed ( ActionEvent event ) {
            for ( int i1 = 0; i1 < fac1.size(); i1++ ) {
                bFac[i1][0].setBackground( null );
                if ( event.getActionCommand().equals( fac1.get( i1 ) ) ) {
                    i = i1;
```

```
                    bFac[il][0].setBackground( Color.green );
                }
            }
        }
    }

    private class ButtonHandlerTo implements ActionListener {
        public void actionPerformed ( ActionEvent event ) {
            for ( int il = 0; il < fac1.size(); il++ ) {
                bFac[il][1].setBackground( null );
                if ( event.getActionCommand().equals( fac1.get(il)) ) {
                    j = il;
                    bFac[il][1].setBackground( Color.green );
                }
            }
        }
    }

    private class OKHandler implements ActionListener {
        public void actionPerformed ( ActionEvent event ) {
            units[i][j][m] = Integer.parseInt( unitsText.getText());
            jTravel.setVisible( false );
        }
    }

}
```

Distance class

Distance
- l[ ][ ][ ]: Integer - place[ ][ ][ ] : Integer - l, j, t, m, t1[ ]: Integer - M: integer = 999999 - type[ ]: String = {“road”,”rail”,”pipeline”,”conveyer”,”waterway”} - b: boolean - f[ ][ ][ ]: boolean - place1[ ][ ][ ]: boolean - chi[ ][ ][ ]:boolean - placeC[ ][ ][ ]: boolean - b1: boolean - fac1: ArrayList <String> - travelCosts1: TravelCosts
+ setD(fac: ArrayList <String>, materials: Materials, travel Travel) + setDistance(fac: ArrayList <String>, Materials: materials, Travel: travel) + defineL(fac: ArrayList <String>) + getL(): Integer + getLargeNumber(): Integer + getLa(): integer[ ][ ][ ] + getTravelCosts(): TravelCosts + getF(): boolean + geti(): integer + getj(): integer + getFa(): boolean[ ][ ][ ] + defineF(fac: ArrayList <String>) + getUEa(): long[ ][ ][ ] + getFCa(): long[ ][ ][ ] + getFVa(): long[ ][ ][ ] + getUNa(): long[ ][ ] **+ execute()**

Figure35: Distance UML class diagram

Travel distances between departments are defined using this class. It also is the launch pad and temporary storage for the class TravelCosts.

```
package logistics;

/*  Sasol and University of Pretoria
 *  Student Number 25245679
 *  Name Jakus van Rooyen
 */

import java.awt.Color;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
```

```
public class Distance {

    private int l[][][], place[][][], i, j,t,m, tl[],ml[], M = 999999; // M is a large number
    private String type[] = {"road","rail","pipeline","conveyer","waterway"};
    private boolean b;
    private boolean f[][][], placel[][][], chi[][][], placeC[][][], bl;
    private long ue[][][], fc[][][], fv[][][],un[][][];

    private JFrame jD, jDis;
    private JLabel jLabel, jLabel2;
    private JPanel jPanel;
    private JButton bFac[][], OK1, OK;
    private JTextField tText[];
    private JCheckBox avail[];

    private int counter;
    private Materials materials1;
    private Travel travel1;
    private TravelCosts travelCosts1;
    private List <String> fac1;

    public Distance() {
        travelCosts1 = new TravelCosts();
    }

    public void setD( List<String> fac, Materials materials, Travel travel ) {

        materials1 = materials;
        travel1 = travel;
        fac1 = fac;
        jD = new JFrame( "Between facilities" );
        defineL( fac );
        jD.setLayout( new GridLayout( 4 + fac1.size() ,1,3,3 ) );
        jD.setVisible( true );
        jD.setDefaultCloseOperation( JFrame.HIDE_ON_CLOSE );
        jD.setSize( 300, 4*50 + fac1.size()*50 );
        jLabel = new JLabel( "Select the supply and demand facility" );
        jD.add( jLabel );
        jLabel = new JLabel( "to specify distances between" );
        jD.add( jLabel );
        jLabel = new JLabel( "From facility", SwingConstants.CENTER );
        jLabel2 = new JLabel( "To facility", SwingConstants.CENTER );
        jPanel = new JPanel();
        jPanel.add( jLabel );
        jPanel.add( jLabel2 );
        jD.add( jPanel);
        bFac = new JButton[fac.size()][2];
        ButtonHandlerTo handlerTo = new ButtonHandlerTo();
        ButtonHandlerFrom handlerFrom = new ButtonHandlerFrom();
        for ( int i1 = 0; i1 < fac.size(); i1++ ) {
            jPanel = new JPanel();
            jPanel.setLayout( new GridLayout( 1, 2) );
            bFac[i1][0] = new JButton( fac.get( i1 ) );
            bFac[i1][1] = new JButton( fac.get( i1 ) );
            bFac[i1][0].addActionListener( handlerFrom );
            bFac[i1][1].addActionListener( handlerTo );
            jPanel.add( bFac[i1][0] );
            jPanel.add( bFac[i1][1] );
            jD.add( jPanel );
        }
        OK = new JButton ( "OK" );
        OKHandler oKHandler = new OKHandler();
        OK.addActionListener( oKHandler );
        jD.add( OK );
    }

    public void setDistance( List<String> fac, Materials materials, Travel travel ) {
        int counterX;
        jDis = new JFrame ( "Set distances" );
        jDis.setVisible( false );
        jLabel = new JLabel( "Set Distances from " + fac.get( i ) +  " to " + fac.get( j ) );
        jDis.add( jLabel );
        counterX = 1;
        counter = 0;
        for ( m = 0; m < materials.getMat().size(); m++ ) for ( t = 0; t < 5; t++ ) {
            if ( ( materials.getMT( m, t ) ) && ( l[i][j][t] == 0 ) && ( travel.getV( i, j, m ) > 0 ) && !chi[i][j][t] ) {
                counter++;
                chi[i][j][t] = true;

            }
        }

        chi = new boolean[fac.size()][fac.size()][5];
        tText = new JTextField[counter];
        tl = new int[counter];
        ml = new int[counter];

        avail = new JCheckBox[counter];
        counter = 0;
        defineF( fac );
        for ( m = 0; m < materials.getMat().size(); m++ ) for ( t = 0; t < 5; t++ ) {
            if ( ( materials.getMT( m, t ) ) && ( l[i][j][t] == 0 ) && ( travel.getV( i, j, m ) > 0 ) && ( !chi[i][j][t] ) ) {
                jLabel = new JLabel ( "for material " + materials.getMat().get( m ) + " using travel method " + type[t] );
                jDis.add( jLabel );
                jPanel = new JPanel();
                jPanel.setLayout( new GridLayout( 1, 2 ) );
                jLabel = new JLabel( "infrastructure available? " );
                jPanel.add( jLabel );
                avail[counter] = new JCheckBox( "Available" );
                jPanel.add( avail[counter] );
                jDis.add( jPanel );
                jPanel = new JPanel();
                jPanel.setLayout( new GridLayout( 1 ,2 ) );
                jLabel = new JLabel( "add Distances" );
                jPanel.add( jLabel );
                tText[counter] = new JTextField( "", 15 );
                jPanel.add( tText[counter] );
                jDis.add( jPanel );
                counterX = counterX + 3;
                tl[counter] = t;
                counter++;
                chi[i][j][t] = true;
            }
        }

        OKHandler1 handler1 = new OKHandler1();

        OK1 = new JButton( "OK" );
        OK1.addActionListener( handler1 );
        jDis.add( OK1 );
        counterX = counterX + 1;
        int gf = 0;
```

```
        if ( counterX*50 < 700 ) gf = counterX*50; else  gf = 700;
        jDis.setSize( 365, gf );
        jDis.setLayout( new GridLayout( counterX, 1, 3, 3 ) );
        jDis.setVisible( true );
        jDis.setDefaultCloseOperation( JFrame.HIDE_ON_CLOSE );
    }


    private void defineL( List<String> fac ) {
        if ( b == false ) {
            l = new int[fac.size()][fac.size()][5];
        }
        else {
            place = new int[fac.size()][fac.size()][5];
            for ( int counter1 = 0; counter1 < l.length; counter1++ ) {
                for ( int counter2 = 0; counter2 < l[counter1].length; counter2++ ) {
                    for ( int counter3 = 0; counter3 < 5; counter3++ ) {
                        place[counter1][counter2][counter3] = l[counter1][counter2][counter3];
                    }
                }
            }
            l = new int[fac.size()][fac.size()][5];
            l = place;
        }
        b = true;
    }

    public int getL( int i1, int i2, int i3 ) {
        return l[i1][i2][i3];
    }

    public int getLargeNumber() {
        return M;
    }

    public int[][][] getLa() {
        return l;
    }

    public TravelCosts getTravelCosts() {
        return travelCosts1;
    }

    private class ButtonHandlerFrom implements ActionListener {
        public void actionPerformed ( ActionEvent event ) {
            for ( int o = 0; o < fac1.size(); o++ ) {
                bFac[o][0].setBackground( null );
                if ( event.getActionCommand().equals( fac1.get( o ) ) ) {
                    i = o;
                    bFac[o][0].setBackground( Color.green );
                }
            }
        }
    }

    private class ButtonHandlerTo implements ActionListener {
        public void actionPerformed ( ActionEvent event ) {
            for ( int o = 0; o < fac1.size(); o++ ) {
                bFac[o][1].setBackground( null );
                if ( event.getActionCommand().equals( fac1.get(o)) ) {
                    j = o;
                    bFac[o][1].setBackground( Color.green );
                }
            }
        }
    }

    private class OKHandler implements ActionListener {
        public void actionPerformed ( ActionEvent event ) {
            defineL( fac1 );
            defineF( fac1 );
            setDistance( fac1, materials1, travel1 );
            jD.setVisible( false );
            jDis.setVisible( true );
            for ( int cou1 = 0; cou1 < bFac.length; cou1++ ) {
                for ( int cou2 = 0; cou2 < bFac[cou1].length; cou2++ ) {
                    bFac[cou1][cou2].setBackground( null );
                }
            }
        }
    }

    private class OKHandler1 implements ActionListener {
        public void actionPerformed ( ActionEvent event ) {
            for ( counter = 0; counter < tl.length; counter++ ) {
                l[i][j][tl[counter]] = Integer.parseInt( tText[counter].getText() );
                if ( avail[counter].isSelected() ) {
                    f[i][j][tl[counter]] = true;
                }
                else {
                    f[i][j][tl[counter]] = false;
                }
            }
            travelCosts1.setICost( i, j, f, fac1, materials1, travel1 );
            jDis.setVisible( false );

        }
    }

//**********Start avail*************************************************


    public boolean getF( int i1, int i2, int i3 ) {
        return f[i1][i2][i3];
    }

    public int geti() {
        return i;
    }

    public int getj() {
        return j;
    }

    public boolean[][][] getFa() {
        return f;
    }

    private void defineF( List<String> fac ) {
        if ( b1 == false ) {
            f = new boolean[fac.size()][fac.size()][5];
            chi = new boolean[fac.size()][fac.size()][5];
        }
```

```
        else {
            place1 = new boolean[fac.size()][fac.size()][5];
            placeC = new boolean[fac.size()][fac.size()][5];
            for ( int i1 = 0; i1 < f.length; i1++ ) {
                for ( int i2 = 0; i2 < f[i1].length; i2++ ) {
                    for ( int i3 = 0; i3 < 5; i3++ ) {
                        place1[i1][i2][i3] = f[i1][i2][i3];
                        placeC[i1][i2][i3] = chi[i1][i2][i3];
                    }
                }
            }
            f = new boolean[fac.size()][fac.size()][5];
            chi = new boolean[fac.size()][fac.size()][5];
            f = place1;
            chi = placeC;
        }
        b1 = true;
    }

    public long[][][] getUEa() {
        return travelCosts1.getUEa();
    }
    public long[][][] getFCa() {
        return travelCosts1.getFCa();
    }
    public long[][][] getFVa() {
        return travelCosts1.getFVa();
    }
    public long[][][] getUNa() {
        return  travelCosts1.getUNa();
    }


}
```

TravelCosts class

TravelCosts
- ue[ ][ ][ ]: long - fv[ ][ ][ ]: long - fc[ ][ ][ ]: long - un[ ][ ][ ]: long - I,j: integer - b: boolean - f[ ][ ][ ]: boolean - chi[ ][ ][ ]: boolean - Ttype[ ]: String = {“road”,”rail”,”pipeline”,”conveyer”,”waterway”}
+ setICost(i: integer, j: integer, f[ ][ ][ ]: boolean, fac: ArrayList <String>, materials: Materials, travel: Travel) + defineArrays(fac: ArrayList <String>) + getUE(i1: integer, i2: integer, i3: integer): long + getFC(i1: integer, i2: integer, i3: integer): long + getFV(i1: integer, i2: integer, i3: integer): long + getUN(i1: integer, i2: integer, i3: integer): long + getUEa(): long[ ][ ][ ] +getFCa(): long[ ][ ][ ] +getFVa(): long[ ][ ][ ] + getUNa(): long[ ][ ][ ]

Figure36: TravelCosts UML class diagram

All travel costs are done here, and are launched just after the distances are inputted between two specific facilities.

```
package logistics;

/*  Sasol and University of Pretoria
 *  Student Number 25245679
 *  Name Jakus van Rooyen
 */

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;

public class TravelCosts {

    private long ue[][][],fc[][][], fv[][][],un[][][];
    private long placeUE[][][],placeUN[][][], placeFC[][][],placeFV[][][];
    private int i,j;
    private boolean b , f[][][], chi[];
    private String Ttype[] = { "road", "rail", "pipeline", "conveyer", "waterway" };
    private JFrame jFrame;
    private JLabel jLabel;
    private JTextField ueT[], fcT[], fvT[], unT[];
    private JButton OK;

    public void setICost ( int i1, int i2, boolean f1[][][], List<String> fac, Materials materials, Travel travel ) {
        i = i1;
        j = i2;
        f = f1;
        int counter = 0;
        jFrame = new JFrame( "Infrastructure costs" );
```

```
        jFrame.setVisible( false );
        jFrame.setDefaultCloseOperation( JFrame.HIDE_ON_CLOSE );
        defineArrays( fac );
        chi = new boolean[5];
        ueT = new JTextField[5];
        fcT = new JTextField[5];
        fvT = new JTextField[5];
        unT = new JTextField[5];
        for ( int t = 0; t < 5; t++ ) {
            ueT[t] = new JTextField( "" );
            fcT[t] = new JTextField( "" );
            fvT[t] = new JTextField( "" );
            unT[t] = new JTextField( "" );
        }

        for ( int m = 0; m < materials.getMat().size(); m++ ) {
            for ( int t = 0; t < 5; t++ ) {
                if ( ( f[i][j][t] ) && materials.getMT( m, t ) && ( travel.getV(i, j, m) > 0 ) && ( ue[i][j][t] == 0 ) && !chi[t] ) {
                    jLabel = new JLabel( "Please enter OPEX COST per unit.distance" );
                    jFrame.add( jLabel );
                    jLabel = new JLabel( "for " + Ttype[t] + ":");
                    jFrame.add( jLabel );
                    jFrame.add( ueT[t] );
                    counter = counter + 3;
                    chi[t] = true;
                }
                else if ( ( !f[i][j][t] ) && materials.getMT( m, t ) && ( travel.getV( i, j, m ) > 0 ) ) {
                    if (fc[i][j][t] == 0 ) {
                        jLabel = new JLabel( "Capex and Opex for infrstucture Unavailable" );
                        jFrame.add( jLabel );
                        jLabel = new JLabel( "Please enter capex fixed cost to build " + Ttype[t] );
                        jFrame.add( jLabel );
                        jFrame.add( fcT[t] );
                        jLabel = new JLabel( "Please enter capex cost per unit distance for" + Ttype[t]  );
                        jFrame.add( jLabel );
                        jFrame.add( fvT[t] );
                        jLabel = new JLabel( "Please enter opex cost per unit distance for " + Ttype[t] );
                        jFrame.add( jLabel );
                        jFrame.add( unT[t] );
                        chi[t] = true;
                        counter = counter + 7;
                    }
                }
            }
        }
        OK = new JButton( "OK" );
        OKHandler oKHandler = new OKHandler();
        OK.addActionListener( oKHandler );
        jFrame.add( OK );
        counter++;
        jFrame.setLayout( new GridLayout( counter, 1 ) );
        int gs;
        if ( counter*50 < 700 ) gs = counter*50; else gs = 700;
        jFrame.setSize( 300, gs );
        jFrame.setVisible( true );

    }

    private void defineArrays ( List<String> fac) {

        if ( b == false ) {
            fc = new long[fac.size()][fac.size()][5];
            fv = new long[fac.size()][fac.size()][5];
            un = new long[fac.size()][fac.size()][5];
            ue = new long[fac.size()][fac.size()][5];
        }
        else {
            /*
            placeFC = new long[fac.size()][fac.size()][5];
            placeUE = new long[fac.size()][fac.size()][5];
            placeFV = new long[fac.size()][fac.size()][5];
            placeUN = new long[fac.size()][fac.size()][5];
            for ( int counter1 = 0; counter1 < fc.length; counter1++ ) {
                for ( int counter2 = 0; counter2 < fc.length; counter2++ ) {
                    for ( int counter3 = 0; counter3 < 5; counter3++ ) {
                        placeFC[counter1][counter2][counter3] = fc[counter1][counter2][counter3];
                        placeUE[counter1][counter2][counter3] = ue[counter1][counter2][counter3];
                        placeFV[counter1][counter2][counter3] = fv[counter1][counter2][counter3];
                        placeUN[counter1][counter2][counter3] = un[counter1][counter2][counter3];
                    }
                }
            }
            fc = new long[fac.size()][fac.size()][5];
            fv = new long[fac.size()][fac.size()][5];
            un = new long[fac.size()][fac.size()][5];
            ue = new long[fac.size()][fac.size()][5];
            fc = placeFC;
            fv = placeFV;
            un = placeUN;
            ue = placeUE;
            */
        }
        b = true;
    }



    public long getUE( int i1, int i2, int i3 ) {
        return ue[i1][i2][i3];
    }

    public long getFC( int i1, int i2, int i3 ) {
        return fc[i1][i2][i3];
    }


    public long getFV( int i1, int i2, int i3 ) {
        return fv[i1][i2][i3];
    }

    public long getUN( int i1, int i2, int i3 ) {
        return un[i1][i2][i3];
    }

    public long[][][] getUEa() {
        return ue;
    }
    public long[][][] getFCa() {
        return fc;
    }
    public long[][][] getFVa() {
        return fv;
```

```
    }
    public long[][][] getUNa() {
        return un;
    }
    private class OKHandler implements ActionListener {
        public void actionPerformed ( ActionEvent event ) {
            for ( int t = 0; t < 5; t++ ) {
                if ( ( ue[i][j][t] == 0 ) && ( !ueT[t].getText().equals( "" ) ) ) ue[i][j][t] = Long.parseLong( ueT[t].getText() );
                if ( ( fc[i][j][t] == 0 ) && ( !fcT[t].getText().equals( "" ) ) ) fc[i][j][t] = Long.parseLong( fcT[t].getText() );
                if ( ( fv[i][j][t] == 0 ) && ( !fvT[t].getText().equals( "" ) ) ) fv[i][j][t] = Long.parseLong( fvT[t].getText() );
                if ( ( un[i][j][t] == 0 ) && ( !unT[t].getText().equals( "" ) ) ) un[i][j][t] = Long.parseLong( unT[t].getText() );
            }
            jFrame.setVisible( false );
        }
    }
}
```

Utility classes

Record class

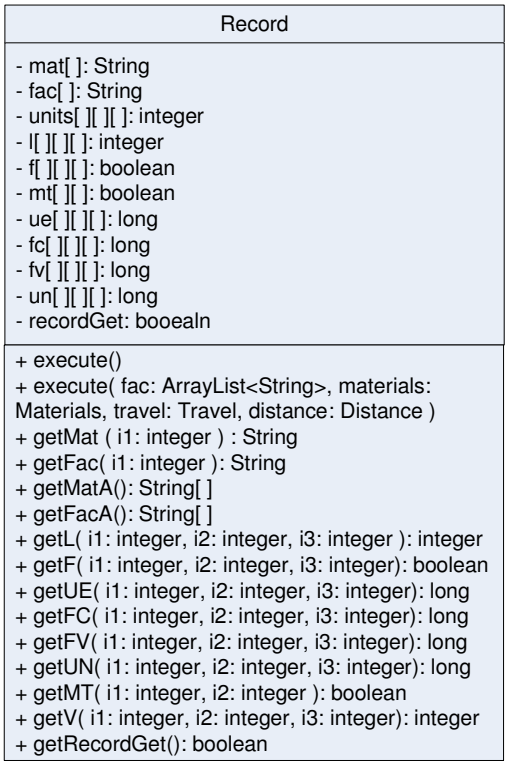


Figure37: Record UML class diagram

All inputs are stored permanently in Record class and all calculations are done out of record class.

```
package logistics;

/*  Sasol and University of Pretoria
 *  Student Number 25245679
 *  Name Jakus van Rooyen
 */

import java.io.Serializable;
import java.util.List;


public class Record implements Serializable {

    private String[] matA, facA;
    private int units[][][], l[][][];
    private boolean f[][][], mt[][];
    private long ue[][][],fc[][][], fv[][][],un[][][];
    private Boolean recordGet;

    public Record () {
    }

    public Record ( List<String> fac, Materials materials, Travel travel, Distance distance ) {
        recordGet = true;
        units = travel.getVa();
        l = distance.getLa();
        f = distance.getFa();
        ue = distance.getUEa();
        fc = distance.getFCa();
        fv = distance.getFVa();
        un = distance.getUNa();
        mt = materials.getMTa();
        matA = new String[materials.getMat().size()];
        facA = new String[fac.size()];

        for ( int counter1 = 0; counter1 < materials.getMat().size(); counter1++ ) {
            matA[counter1] = materials.getMat().get( counter1 );
        }
        for ( int counter1 = 0; counter1 < fac.size(); counter1++ ) {
```

```
            facA[counter1] = fac.get( counter1 );
        }
    }

    public String getMat ( int i1 ) {
        return matA[i1];
    }

    public String getFac ( int i1 ) {
        return facA[i1];
    }

    public String[] getMatA() {
        return matA;
    }

    public String[] getFacA() {
        return facA;
    }

    public int getL( int i1, int i2, int i3 ) {
        return l[i1][i2][i3];
    }

    public boolean getF( int i1, int i2, int i3 ) {
        return f[i1][i2][i3];
    }

    public long getUE( int i1, int i2, int i3 ) {
        return ue[i1][i2][i3];
    }

    public long getFC( int i1, int i2, int i3 ) {
        return fc[i1][i2][i3];
    }


    public long getFV( int i1, int i2, int i3 ) {
        return fv[i1][i2][i3];
    }

    public long getUN( int i1, int i2, int i3 ) {
        return un[i1][i2][i3];
    }

    public boolean getMT ( int  i1, int i2 ) {
        return mt[i1][i2];
    }

    public int getV( int i1, int i2, int i3 ) {
        return units[i1][i2][i3];
    }

    public Boolean getRecordGet() {
        return recordGet;
    }


}
```

RecordManager class

RecordManager
- output: ObjectOutputStream - input: ObjectInputStream - record: Record - fileName: String
- openNewFile() - saveFile(record: Record) - openExistingFile(): Record - addRecord(record: Record) - closeFile()

Figure38: RecordManager UML class diagram

New logistics chain records are created, saved, opened, and closed here.

```
package logistics;

/*  Sasol and University of Pretoria
 *  Student Number 25245679
 *  Name Jakus van Rooyen
 */

import java.io.EOFException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.NoSuchElementException;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JOptionPane;


public class RecordManager extends JFrame {
    private ObjectOutputStream output;
    private ObjectInputStream input;



    public void openNewFile() {
            output = null;
            record = null;
    }

    private Record record;
```

```
    private String fileName;

    public void saveFile( Record record1 ) {
        try {
            if ( output == null ) {
                fileName = JOptionPane.showInputDialog( "Please enter file name to save" );
                output = new ObjectOutputStream( new FileOutputStream( fileName + ".ser" ) );
            }
            addRecord( record1 );
        }
        catch ( IOException ioException ){
            JOptionPane.showMessageDialog( null , "Error creating file." );
        }
    }

    public Record openExistingFile() {
        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setFileSelectionMode( JFileChooser.FILES_AND_DIRECTORIES );
        int result = fileChooser.showOpenDialog( this );
        // if user clicked Cancel button on dialog, return

        File fileName1 = fileChooser.getSelectedFile();


        if ( ( fileName1 == null ) || ( fileName1.getName().equals( "" ) ) ) {
            JOptionPane.showMessageDialog( this, "Invalid File Name",
            "Invalid File Name", JOptionPane.ERROR_MESSAGE );
            System.exit( 1 );
        } // end if
        try {
            input = new ObjectInputStream( new FileInputStream( fileName1.getName() ) );
        } // end try
        catch ( IOException ioException ) {
            JOptionPane.showMessageDialog( null , "Error opening file." );
        } // end catch

        try {
            record = ( Record ) input.readObject();
            return record;
        }
        catch ( EOFException endOfFileException ) {
            JOptionPane.showMessageDialog( null , "No records in file" );
            System.exit( 1 );
            record = new Record();
            return record;
        }
        catch ( ClassNotFoundException classNotFoundException ) {
            JOptionPane.showMessageDialog( null , "Unable to create object" );
            System.exit( 1 );
            record = new Record();
            return record;
        }
        catch ( IOException ioException ) {
            JOptionPane.showMessageDialog( null , "Error during reading from file" );
            record = new Record();
            return record;
        }
    }

    public void addRecord( Record record ) {

        try {
            output.writeObject( record );
            JOptionPane.showMessageDialog( null , "Saved file in program folder under " + fileName + ".ser" );
        }
        catch ( IOException ioException ) {
            JOptionPane.showMessageDialog( null , "Error writing to file." );
            return;
        }
        catch (NoSuchElementException elementException ) {
            JOptionPane.showMessageDialog( null , "Invalid input. Please try again." );
            return;
        }
    }

    public void closeFile() {
        try{
            if ( output != null ) {
                output.close();
            }
            if ( input != null ) {
                input.close();
            }
        }
        catch ( IOException ioException ) {
            JOptionPane.showMessageDialog( null , "Error closing file(s)" );
            System.exit( 1 );
        }
    }
}
```

Calculate and output class

Calculate class

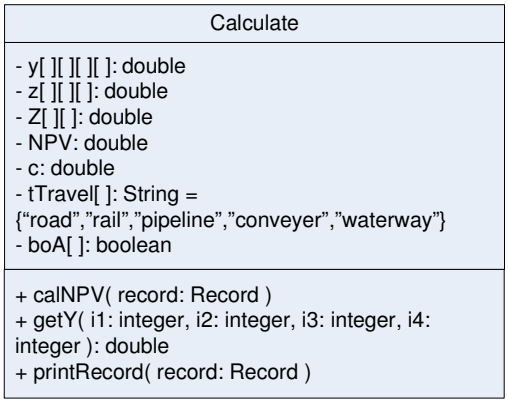


Figure39: Calculate UML class diagram

All calculations and outputs occur using this class.

```
package logistics;

/*  Sasol and University of Pretoria
 *  Student Number 25245679
 *  Name Jakus van Rooyen
 */

import java.awt.GridLayout;
import java.io.FileNotFoundException;
import java.util.Formatter;
import java.util.FormatterClosedException;
import java.util.NoSuchElementException;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;


public class Calculate {

    private double y[][][][], z[][][], Z[][],  NPV, c;
    private String tTravel[] = { "road", "rail", "pipeline", "conveyer", "waterway" };
    private boolean boA[];

    public void calNPV( Record record ) {

        c = 33.3333333333333333;
        y = new double[record.getFacA().length][record.getFacA().length][record.getMatA().length][5];
        z = new double[record.getFacA().length][record.getFacA().length][record.getMatA().length];
        Z = new double[record.getFacA().length][record.getFacA().length];
        double M;
        M = 0;
        for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
            for( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                for( int m = 0; m < record.getMatA().length; m++ ) {
                    for( int t = 0; t < 5; t++ ) {
                        if ( ( record.getV( f1, f2, m ) > 0 ) && record.getMT( m, t ) )  {
                            if ( record.getF( f1, f2, t ) ) {
                                y[f1][f2][m][t] = record.getV( f1, f2, m ) * record.getUE( f1, f2, t ) * record.getL(f1, f2, t) * c;
                            }
                            else {
                                y[f1][f2][m][t] = record.getV( f1, f2, m )* record.getUN(f1, f2, t ) * record.getL(f1, f2, t )* c +
record.getFC(f1, f2, t ) + record.getFV(f1, f2, t ) * record.getL(f1, f2, t );
                            }
                        }

                        else {
                            y[f1][f2][m][t] = 0;
                        }
                        if ( y[f1][f2][m][t] > M ) {
                            M = y[f1][f2][m][t] * 2;
                        }
                    }
                    z[f1][f2][m] = M;
                    for( int t = 0; t < 5; t++ ) {
                        if ( ( z[f1][f2][m] > y[f1][f2][m][t] ) && ( y[f1][f2][m][t] != 0 ) ) {
                            z[f1][f2][m] = y[f1][f2][m][t];
                        }
                    }
                    if ( z[f1][f2][m] == M ) {
                        z[f1][f2][m] = 0;
                    }
                }
                Z[f1][f2] = 0;
                for ( int m = 0; m < record.getMatA().length; m++ ) {
                    Z[f1][f2] = Z[f1][f2] + z[f1][f2][m];
                }
            }
        }

        NPV = 0;
        for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
            for( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                NPV = NPV + Z[f1][f2];
            }
        }
```

```
}

public double getY( int i1, int i2, int i3, int i4 ) {
    return y[i1][i2][i3][i4];
}

Formatter input;

public void printTextRecord ( Record record ) {
    int c1 = 0;
    boolean print;
    String s;
    print = false;


    s = JOptionPane.showInputDialog( "Please enter text file name" );
    s = s + ".txt";
    try {
        input = new Formatter( s );
    }
    catch ( SecurityException secirityException ) {
        JOptionPane.showMessageDialog( null, "You do not have write access to this file." );
    }
    catch ( FileNotFoundException filesNotFoundException ) {
        JOptionPane.showMessageDialog( null, "Error creating file." );
    }
    try {
        for ( int counter1 = 0; counter1 < record.getMatA().length; counter1++ ) {
            s = "MATERIAL: " + record.getMat(counter1) + " USING METHOD: ";
            for ( int counter2 = 0; counter2 < 5; counter2++ ) {
                if ( record.getMT( counter1, counter2 ) == true) {
                    s = s + tTravel[counter2] + " ";
                }
            }
            input.format( "%s\n", s );
        }
        s = "FACILITIES: ";
        for ( int counter1 = 0; counter1 < record.getFacA().length; counter1++ ) {
            s = s + record.getFac(counter1) + " ";
        }
        input.format( "%s\n", s );
        input.format( "%s\n", "INFRASTRUCTURE AVAILABILITY:" );
        for ( int t = 0; t < 5; t++ ) {
            print = false;
            for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                    if ( record.getF( f1, f2, t ) ) {
                        print = true;
                    }
                }
            }
            if ( print ) {
                input.format( "%s\n","FOR TRAVEL METHOD " + tTravel[t] + ":" );

                s = String.format( "%-30s", "");
                for ( int counter1 = 0; counter1 < record.getFacA().length; counter1++ ) {
                    s = s + String.format( "%-30s", record.getFac(counter1) );
                }
                input.format( "%s\n", s );
                s = "";
                for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                    s = String.format( "%-30s", record.getFac( f1 ) );
                    for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                        s = s + String.format( "%-30s", record.getF( f1, f2, t ) );
                    }
                    input.format( "%s\n", s );
                }
            }
        }

        input.format( "%s\n","UNITS BETWEEN FACILITIES: " );
        for ( int counter1 = 0; counter1 < record.getMatA().length; counter1++ ) {
            print = false;
            for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                    if ( record.getV( f1, f2, counter1) > 0 ) {
                        print = true;
                    }
                }
            }
            if ( print ) {
                s = "FOR MATERIAL: ";
                s = s + record.getMat(counter1);
                input.format( "%s\n", s );
                s = "";
                s =String.format( "%-30s", "");
                for ( int p = 0; p < record.getFacA().length; p++ ) s = s + String.format( "%-30s", record.getFac(p) );
                input.format( "%s\n", s );
                for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {

                    s = "";
                    s = String.format( "%-30s", record.getFac(f1) );
                    for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                        s = s + String.format( "%-30d", record.getV( f1, f2, counter1) );
                    }
                    input.format( "%s\n", s );
                }
            }
        }


        for ( int t = 0; t < 5; t++ ) {
            print = false;
            for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                    if ( record.getUE( f1, f2, t ) > 0 ) {
                        print = true;
                    }
                }
            }
            if ( print ) {
                input.format( "%s %s\n", tTravel[t] , "COST PER UNIT.DISTANCE ON AVAILABILE INFRASTRUCTURE:" );
                s = String.format( "%-30s", "");
                for ( int p = 0; p < record.getFacA().length; p++ ) s = s + String.format( "%-30s", record.getFac(p) );
                input.format( "%s\n", s );
                s = "";
                for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                    s = String.format( "%-30s", record.getFac(f1) );
                    for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                        s = s + String.format( "%-30d", record.getUE( f1, f2, t) );
                    }
                    input.format( "%s\n", s );
```

```
                    }
                }
            }
            for ( int t = 0; t < 5; t++ ) {
                print = false;
                for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                    for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                        if ( ( record.getFC( f1, f2, t ) > 0 ) || ( record.getFV( f1, f2, t ) > 0 ) || ( record.getUN( f1, f2, t ) > 0 ) )
{
                            print = true;
                        }
                    }
                }
                if ( print ) {

                    input.format( "%s %s\n", tTravel[t], "FIXED CAPEX COST ON UNAVAILABLE INFRASTRUCTURE:" );

                    s = String.format( "%-30s", "");
                    for ( int p = 0; p < record.getFacA().length; p++ ) s = s + String.format( "%-30s", record.getFac(p) );
                    input.format( "%s\n", s );
                    for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                        s = String.format( "%-30s", record.getFac(f1) );
                        for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                            s = s + String.format( "%-30d", record.getFC( f1, f2, t ) );
                        }
                        input.format( "%s\n", s );
                    }
                    print = false;
                    for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                        for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                            if ( record.getFV( f1, f2, t ) > 0 ) {
                                print = true;
                            }
                        }
                    }
                    if ( print ) {
                        input.format( "%s %s\n", tTravel[t], "CAPEX COST PER UNIT.DISTANCE ON UNAVAILABLE INFRASTRUCTURE:" );
                        s = String.format( "%-30s", "");
                        for ( int p = 0; p < record.getFacA().length; p++ ) s = s + String.format( "%-30s", record.getFac(p) );
                        input.format( "%s\n", s );
                        for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {

                            s = String.format( "%-30s", record.getFac(f1) );
                            for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                                s = s + String.format( "%-30d", record.getFV( f1, f2, t ) );
                            }
                            input.format( "%s\n", s );
                        }
                    }
                    print = false;
                    for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                        for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                            if ( record.getFC( f1, f2, t ) > 0 ) {
                                print = true;
                            }
                        }
                    }
                    if ( print ) {
                        input.format( "%s %s\n", tTravel[t], "OPEX COST PER UNIT.DISTANCE ON UNAVAILABLE INFRASTRUCTURE:" );
                        s = String.format( "%-30s", "");
                        for ( int p = 0; p < record.getFacA().length; p++ ) s = s + String.format( "%-30s", record.getFac(p) );
                        input.format( "%s\n", s );
                        for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                            s = String.format( "%-30s", record.getFac(f1) );
                            for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                                s = s + String.format( "%-30d", record.getUN( f1, f2, t ) );
                            }
                            input.format( "%s\n", s );
                        }
                    }
                }
            }

            input.format( "%s\n", "DISTANCES BETWEEN FACILITIES:" );
            for ( int t = 0; t < 5; t++ ) {
                print = false;
                for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                    for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                        if ( record.getL( f1, f2, t) > 0 ) {
                            print = true;
                        }
                    }
                }
                if ( print ) {
                    s = String.format( "FOR TRAVEL METHOD %s:", tTravel[t] );
                    input.format( "%s\n", s );
                    s = String.format( "%-30s", "" );
                    for ( int counter = 0; counter < record.getFacA().length; counter++ ) {
                        s = s + String.format( "%-30s", record.getFac( counter ) );
                    }
                    input.format( "%s\n", s );
                    for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                        s = String.format( "%-30s", record.getFac( f1 ) );
                        for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                            s = s + String.format( "%-30d", record.getL( f1, f2, t ) );
                        }
                        input.format( "%s\n", s );
                    }
                }
            }
            input.format( "%s\n", "for 'y':" );
            for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                for( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                    for( int m = 0; m < record.getMatA().length; m++ ) {
                        for( int t = 0; t < 5; t++ ) {
                            if ( y[f1][f2][m][t] > 0 ) {
                                s = String.format( "%.2f", y[f1][f2][m][t] );
                                s = s + String.format( ": <from fac %s><to fac %s><material %s><transport %s>\n", record.getFac( f1 ),
record.getFac( f2 ), record.getMat( m ), tTravel[t] );
                                input.format( "%s\n", s );
                            }
                        }
                    }
                }
            }
            input.format( "%s\n", "for 'z':" );
            for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                for( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                    for( int m = 0; m < record.getMatA().length; m++ ) {
                        if ( z[f1][f2][m] > 0 ) {
                            s = String.format( "%.2f", z[f1][f2][m] );
                            s = s + String.format( ": <from fac %s><to fac %s><material %s>", record.getFac( f1 ), record.getFac( f2 ),
record.getMat( m ) );
```

```
                        input.format( "%s\n", s );

                    }
                }
            }
        }
        input.format( "%s\n", "for 'Z':" );
        for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
            for( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                if ( Z[f1][f2] > 0 ) {
                    s = String.format( "%.2f", Z[f1][f2] ) + String.format( ": <from fac %s><to fac %s>", record.getFac( f1 ),
record.getFac( f2 ) );
                    input.format( "%s\n", s );
                }
            }
        }
        input.format( "%s\n", "" );
        input.format( "%s\n", "NPV: " + String.format( "%.2f", NPV ) );
    }
    catch ( FormatterClosedException formatterClosedException ) {
        JOptionPane.showMessageDialog( null, "Error writing to file." );
    }
    catch ( NoSuchElementException elementException ) {
        JOptionPane.showMessageDialog( null, "Invalid input. Please try again." );
    }

    if ( input != null )
        input.close();

}


private JFrame jRecord;
private JLabel jLabel;


public void printGUIRecord( Record record ) {
    int c1 = 0;
    boolean print;
    String s = "";
    print = false;
    jRecord = new JFrame( "Record Report" );
    jRecord.setVisible( false );


    for ( int counter1 = 0; counter1 < record.getMatA().length; counter1++ ) {
        s = "MATERIAL: " + record.getMat(counter1) + " USING METHOD: ";
        for ( int counter2 = 0; counter2 < 5; counter2++ ) {
            if ( record.getMT(counter1, counter2) == true) {
                s = s + tTravel[counter2] + " ";
            }
        }
        jLabel =  new JLabel( s );
        jRecord.add( jLabel );
        c1++;
    }
    s = "FACILITIES: ";
    for ( int counter1 = 0; counter1 < record.getFacA().length; counter1++ ) {
        s = s + record.getFac(counter1) + " ";
    }
    jLabel =  new JLabel( s );
    jRecord.add( jLabel );
    c1++;
    jLabel =  new JLabel( "INFRASTRUCTURE AVAILABILITY:" );
    jRecord.add( jLabel );
    c1++;
    for ( int t = 0; t < 5; t++ ) {
        print = false;
        for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
            for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                if ( record.getF( f1, f2, t ) ) {
                    print = true;
                }
            }
        }
        if ( print ) {
            jLabel =  new JLabel( "FOR TRAVEL METHOD " + tTravel[t] + ":" );
            jRecord.add( jLabel );
            c1++;

            s = String.format( "%-30s", "");
            for ( int counter1 = 0; counter1 < record.getFacA().length; counter1++ ) {
                s = s + String.format( "%-30s", record.getFac(counter1) );
            }
            jLabel =  new JLabel( s );
            jRecord.add( jLabel );
            c1++;
            s = "";
            for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                s = String.format( "%-30s", record.getFac( f1 ) );
                for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                    s = s + String.format( "%-30s", record.getF( f1, f2, t ) );
                }
                jLabel =  new JLabel( s );
                jRecord.add( jLabel );
                c1++;
            }
        }
    }
    jLabel =  new JLabel( "UNITS BETWEEN FACILITIES: " );
    jRecord.add( jLabel );
    c1++;
    for ( int counter1 = 0; counter1 < record.getMatA().length; counter1++ ) {
        print = false;
        for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
            for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                if ( record.getV( f1, f2, counter1 ) > 0 ) {
                    print = true;
                }
            }
        }
        if ( print ) {
            s = "FOR MATERIAL: ";
            s = s + record.getMat(counter1);
            jLabel =  new JLabel( s );
            jRecord.add( jLabel );
            c1++;

            s = "";
            s =String.format( "%-30s", "");
            for ( int p = 0; p < record.getFacA().length; p++ ) s = s + String.format( "%-30s", record.getFac(p) );
            jLabel =  new JLabel( s );
            jRecord.add( jLabel );
```

```
            c1++;

            for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {

                s = "";
                s = String.format( "%-30s", record.getFac(f1) );
                for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                    s = s + String.format( "%-30d", record.getV( f1, f2, counterl) );
                }
                jLabel =  new JLabel( s );
                jRecord.add( jLabel );
                c1++;
            }
        }
    }
    for ( int t = 0; t < 5; t++ ) {
        print = false;
        for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
            for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                if ( record.getUE( f1, f2, t ) > 0 ) {
                    print = true;
                }
            }
        }
        if ( print ) {
            jLabel =  new JLabel( "COSTS PER UNIT.DISTANCE ON AVAILABILITY INFRASTRUCTURE:" );
            jRecord.add( jLabel );
            c1++;
            s = String.format( "%-30s", "");
            for ( int p = 0; p < record.getFacA().length; p++ ) s = s + String.format( "%-30s", record.getFac(p) );
            jLabel =  new JLabel( s );
            jRecord.add( jLabel );
            c1++;
            s = "";
            for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                s = String.format( "%-30s", record.getFac(f1) );
                for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                    s = s + String.format( "%-30d", record.getUE( f1, f2, t ) );
                }
                jLabel =  new JLabel( s );
                jRecord.add( jLabel );
                c1++;
            }
        }
        print = false;
        for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
            for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                if ( ( record.getFC( f1, f2, t ) > 0 ) || ( record.getFV( f1, f2, t ) > 0 ) || ( record.getUN( f1, f2, t ) > 0 ) ) {
                    print = true;
                }
            }
        }
        if ( print ) {
            jLabel =  new JLabel( "FIXED CAPEX COST ON UNAVAILABLE INFRASTRUCTURE:" );
            jRecord.add( jLabel );
            c1++;
            s = String.format( "%-30s", "");
            for ( int p = 0; p < record.getFacA().length; p++ ) s = s + String.format( "%-30s", record.getFac(p) );
            jLabel =  new JLabel( s );
            jRecord.add( jLabel );
            c1++;
            for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                s = String.format( "%-30s", record.getFac(f1) );
                for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                    s = s + String.format( "%-30d", record.getFC( f1, f2, t ) );
                }
                jLabel =  new JLabel( s );
                jRecord.add( jLabel );
                c1++;
            }
            print = false;
            for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                    if ( record.getFV( f1, f2, t ) > 0 ) {
                        print = true;
                    }
                }
            }
            if ( print ) {
                jLabel =  new JLabel( "CAPEX COST PER UNIT.DISTANCE ON UNAVAILABLE INFRASTRUCTURE:" );
                jRecord.add( jLabel );
                c1++;
                s = String.format( "%-30s", "");
                for ( int p = 0; p < record.getFacA().length; p++ ) s = s + String.format( "%-30s", record.getFac(p) );
                jLabel =  new JLabel( s );
                jRecord.add( jLabel );
                c1++;
                for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                    s = String.format( "%-30s", record.getFac(f1) );
                    for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                        s = s + String.format( "%-30d", record.getFV( f1, f2, t ) );
                    }
                    jLabel =  new JLabel( s );
                    jRecord.add( jLabel );
                    c1++;
                }
            }
            print = false;
            for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                    if ( record.getFC( f1, f2, t ) > 0 ) {
                        print = true;
                    }
                }
            }
            if ( print ) {
                jLabel =  new JLabel( "OPEX COST PER UNIT.DISTANCE ON UNAVAILABLE INFRASTRUCTURE:" );
                jRecord.add( jLabel );
                c1++;
                s = String.format( "%-30s", "");
                for ( int p = 0; p < record.getFacA().length; p++ ) s = s + String.format( "%-30s", record.getFac(p) );
                jLabel =  new JLabel( s );
                jRecord.add( jLabel );
                c1++;
                for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                    s = String.format( "%-30s", record.getFac(f1) );
                    for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                        s = s + String.format( "%-30d", record.getUN( f1, f2, t) );
                    }
                    jLabel =  new JLabel( s );
                    jRecord.add( jLabel );
                    c1++;
                }
```

```
                }
            }
        }

        jLabel =  new JLabel( "DISTANCES BETWEEN FACILITIES:" );
        jRecord.add( jLabel );
        c1++;
        for ( int t = 0; t < 5; t++ ) {
            print = false;
            for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                    if ( record.getL( f1, f2, t) > 0 ) {
                        print = true;
                    }
                }
            }
            if ( print ) {
                s = String.format( "FOR TRAVEL METHOD %s:", tTravel[t] );
                jLabel =  new JLabel( s );
                jRecord.add( jLabel );
                c1++;
                s = String.format( "%-30s", "" );
                for ( int counter = 0; counter < record.getFacA().length; counter++ ) {
                    s = s + String.format( "%-30s", record.getFac( counter ) );
                }
                jLabel =  new JLabel( s );
                jRecord.add( jLabel );
                c1++;
                for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
                    s = String.format( "%-30s", record.getFac( f1 ) );
                    for ( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                        s = s + String.format( "%-30d", record.getL( f1, f2, t ) );
                    }
                    jLabel =  new JLabel( s );
                    jRecord.add( jLabel );
                    c1++;
                }
            }
        }
        jLabel =  new JLabel( "for 'y':" );
        jRecord.add( jLabel );
        c1++;

        for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
            for( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                for( int m = 0; m < record.getMatA().length; m++ ) {
                    for( int t = 0; t < 5; t++ ) {
                        if ( y[f1][f2][m][t] > 0 ) {
                            s = String.format( "%.2f", y[f1][f2][m][t] );
                            s = s + String.format( ": <from fac %s><to fac %s><material %s><transport %s>\n", record.getFac( f1 ),
record.getFac( f2 ), record.getMat( m ), tTravel[t] );
                            jLabel =  new JLabel( s );
                            jRecord.add( jLabel );
                            c1++;
                        }
                    }
                }
            }
        }
        jLabel =  new JLabel( "for 'z':" );
        jRecord.add( jLabel );
        c1++;
        for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
            for( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                for( int m = 0; m < record.getMatA().length; m++ ) {
                    if ( z[f1][f2][m] > 0 ) {
                        s = String.format( "%.2f", z[f1][f2][m] );
                        s = s + String.format( ": <from fac %s><to fac %s><material %s>", record.getFac( f1 ), record.getFac( f2 ),
record.getMat( m ) );
                        jLabel =  new JLabel( s );
                        jRecord.add( jLabel );
                        c1++;

                    }
                }
            }
        }


        jLabel =  new JLabel( "for 'Z':" );
        jRecord.add( jLabel );
        c1++;

        for ( int f1 = 0; f1 < record.getFacA().length; f1++ ) {
            for( int f2 = 0; f2 < record.getFacA().length; f2++ ) {
                if ( Z[f1][f2] > 0 ) {
                    s = String.format( "%.2f", Z[f1][f2] ) + String.format( ": <from fac %s><to fac %s>", record.getFac( f1 ),
record.getFac( f2 ) );
                    jLabel =  new JLabel( s );
                    jRecord.add( jLabel );
                    c1++;
                }
            }
        }
        jLabel =  new JLabel();
        jRecord.add( jLabel );
        c1++;
        jLabel =  new JLabel( "NPV: " + String.format( "%.2f", NPV ) );
        jRecord.add( jLabel );
        c1++;

        jRecord.setSize( 800, c1*30 );
        jRecord.setDefaultCloseOperation( JFrame.HIDE_ON_CLOSE );
        jRecord.setLayout( new GridLayout( c1 + 1, 1) );
        jRecord.setVisible( true );


    }
}
```

8. Analyse data

The program was tested under expected working conditions. A process of calculating outputs by hand and outputting variables changes were implemented, to serve to check where outputs are misleading or incorrect. A number of types of program issues were identified and solved:

8.1 Data Corruption

8.1.1 Class interface

- Different data types between classes
- Sizes of matrixes defined differently between classes
- Variables have not been initialised

8.1.2 Textbox to variable

- Ability to input the right type of data
- Initialisation of textbox
- Function that converts textbox variable to input variable

8.1.3 Types of data

- Local, global and function variables must be defined correctly
- Variables defined and initialised correctly.
- Data variable sizes must be big enough
- Matrixes must have right amount of dimensions
- Matrix “For”-loops must be defined correctly

8.2 File Saving

8.2.1 Incomplete dataset

- Not all the input data is saved
- Wrong data type

8.2.2 Record to file

- Record not created in file, programming error
- File must be saved under the right name
- *.ser files must be defined correctly

8.3 File and variable management

8.3.1 Variables has been define incorrectly

- Not initialised
- Wrong variable type
- Local and global variables mixed up

8.3.2 Wrong input programming heuristic

- Wrong variables asked for
- Wrong process of information input
- Wrong variables used

8.4 Incomplete output

8.4.1 Not all data shown

- All input information that was used must be shown
- Process of calculation must be shown
- NPV must be shown

8.4.2 Output structure

- Inputs must be shown in logical order (The three steps)
- Columns must be structured
- Variables names must also be shown

8.4.3 Calculations done incorrectly

- Wrong variables used
- For loops not defined correctly
- Criteria not defined correctly
- Variable types does not match
- Steps not in right order
- Inputs not defined correctly
- Mistakes in mathematical formulation
- “for”-loops not correctly defined

8.4.4 Programming done wrong

The IDE (Integrated design environment) picks up the programming that is incorrectly done. Fixing the mistakes requires only a right click, and the Netbeans IDE suggests a correction. These types of problems were not documented.

9. Go over solution

9.1.1 Mathematical verification

Before implementation, a check must be to verify if the programming methods used was within an interval of correctness. Sasol requires that a difference of less the 20% in the programming method and the test method used, for the programming method to be implemented.

I2 supply chain simulation was used as the test method. The author of this report was not involved in checking where that solution found by the programming method was correct. He was told that the first tier was too simplified a version to be implemented, as it does not compensate for supply chain risk, maximum and minimum amounts of materials that can be moved on a given logistics method. And thus the first tier served as foundation for the second tier, to where more depth was added.

The second tier passed. It can now be implemented by Sasol policy.

10. Make a conclusion

Costs in a logistics chain are the costs of moving materials between departments. Once these costs have been identified and calculated, the total logistics cost of a logistics chain scenario can be identified and compared to alternative logistics chain scenarios. This will aid in making a decision as to where a CTL plant can most effectivel y be located.

Appropriate publications were used to define the logistics chain in a manner that warrants the successful completion of the project. The logistics chain was defined in terms of costs associated with logistics. A specific algorithm with an exact solution was the appropriate solution type.

Java was uses in execution, because of its flexibility apposed to Excel. A simple algorithm served as the technique used to calculate a first tier solution. The second tier used the knapsack problem to minimising logistics costs. Extra supply chain information was included in the second tier.

The final design reached its aim of defining logistics chains in terms of logistics costs, together with measurements of storage capacity and supply chain risk. It has been tested under working conditions and improvements were made to the program. Validation was done by simulating the same inputs with a supply chain simulation program called i2. The second tier passed the Sasol implementation test.

This program is accurate enough to predict the costs associated to logistics within a supply chain. It can be used to calculate logistics costs for alternative CTL plant site, and can serve as a tool for selection of the most appropriate CTL plant site.

11. Supplements

11.1 Appendix 1 (Definitions)

Logistics: Defined within this paper as the operation of handling materials from where it is supplied to where it is needed.
Cost: Define within this paper as the NPV equalized cost within a 40 year period and a 20% per year value of money.
Demand: The projected given unit amounts of materials needed annually by a given facility.
Component: A variable or part of the model that has the purpose of creating a more accurate and effective model.

11.3 Appendix 2 (References)

[1] Z. DREZNER, H.W. HAMACHER Eds. 2004. Facility Location: Applications and Theory, Springer, New York
[2] S. NICKEL, J. PUERTO. 2005. Location Theory: A Unified Approach, Springer, New York.
[3] C.S. REVELLE, H.A. EISELT. 2005. Location analysis: A synthesis and survey, European Journal of Operational Research 165 p1–19.
[4] M.T. MELO A.B et al, Facility location and supply chain management – A review.
[5] T. BENDER, H. HENNES et al, Location software and interface with GIS and supply chain management, in: Z. DREZNER, H.W. HAMACHER Eds. 2002 .Facility Location: Applications and Theory, Springer, New York, p. 233–274 (Chapter 8).
[6] L.V. SNYDER.2006. Facility location under uncertainty: A review, IIE Transactions 38, p537–554.
[7] LAWERENCE V. SNYDER: ROSENHEAD et al. 1972, Facility location under uncertainty: a review
[8] Z.-J. SHEN, L. QI. 2007. Incorporating inventory and routing costs in strategic location models, European Journal of Operational Research 179, p372–389
[9] GORDEN C. ARMOUR and ELWOOD S. 1963. A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities, Management Science, Vol. 9, No. 2, p. 294-309
[10] L.V. SNYDER, et al. 2007. The stochastic location model with risk pooling, European Journal of Operational Research 179 p. 1221–1238
[11] D. SIMCHI-LEVI, et al. 2004, Managing the Supply Chain: The Definitive Guide for the Business Professional, McGraw-Hill, New York
[12] J.-F. CORDEAU, et al. 2006. An integrated model for logistics network design, Annals of Operations Research 144 p. 59–82
[13] http://en.wikipedia.org/wiki/Spiral_model, http://www.sasol.com/sasol_internet/frontend/navigation.jsp?rootid=2&navid=2