

https://dspace.org/

The production of software that aids in the design of low voltage distribution networks by optimising the locations of junctions

Apostolellis, Justin

1996

http://hdl.handle.net/10539/22128

Apostolellis, Justin (1996) The production of software that aids in the design of low voltage distribution networks by optimising the locations of junctions, University of the Witwatersrand, Johannesburg,<http://hdl.handle.net/10539/22128>

Downloaded from a DSpace repository. https://dspace.org/

The process of enumeration requires that every permutation is investigated. This is done by initially setting the junction in every junction group to a position that is understood as *unset*. This means that the junction has not been allocated to one of its possible positions. The root junction of the sub-network is then *set* to its cheapest position. All of the sub-networks for which its children are the root junctions are then enumerated. The root junction is then *set* to its next cheapest position.

Inspection of certain *sub-combinations* may reveal the fact that the investigation of all permutations having such a sub-combination should not be made due to problem specific implications that such a sub-combination has. A *sub-combination* refers to the portion of the network, in which the junction's positions have been *set*. Thus all permutations having this sub-combination are implicitly enumerated. These inspections are made according to the criteria for implicit enumeration that have been chosen for the specific problem. Implicit enumeration for this problem is achieved in the following ways:

- CRITERIA 1 : If the cost of all junction groups having junction positions which are *set* is greater than the cost of an entire solution that has already been found, then no improvement on this best solution can be found. Therefore all permutations which have the junction position settings of the junctions whose positions are *set* (of the sub-combination), are implicitly enumerated. The cost of all junctions that are not *set* must be considered in the comparison by adding their individual costs associated with their cheapest positions to the total cost of the junctions whose positions have been *set*.

- CRITERIA 2 : If a particular junction position setting of a sub-combination leads to a violation of the maximum volt drop constraint, then all permutations, having this junction position setting will also lead to a violation and therefore are implicitly enumerated. When the occurrence of a violation of the maximum volt drop constraint is investigated, all junction's whose positions are not set should be placed at the locations that lead to lowest maximum consumer volt drop.

When a permutation is explicitly enumerated, then its total cost and its maximum consumer volt drop are calculated. A solution will eventually be found, otherwise the pre-checking stage would have detected that a solution is impossible.

V. CASE STUDY

A case study is presented showing the application of AJPO to an existing network of a neighbourhood in the Johannesburg area. A representation of the area and its original network are shown in figure 3. The network is a completely underground system, having two transformers and therefore two distributor sub-networks. These two networks are considered separately. The alternative locations that have been selected for each junction are shown in figure 3 by an X. An arrows points from a junction to one of its alternative locations Cable segment routes and lengths corresponding to each junction location were calculated and provided to AJPO as input information.

Figure 4 shows the network configurations found by AJPO. The distributor sub-network on the left has an original cost of R24482 and the optimised network's cost is R22786 thus resulting in a saving of 7 %. The time in which the optimisation was performed was less than one second. The distributor sub-network on the right has an original cost of R25449 and the optimised cost is R24738 thus resulting in a saving of 3%.

The design settings that were used, include an ADMD of 5kW, a breaker current of 30 Amps, the use of 16 mm^2 and 185 mm^2 cables, and a nominal voltage of 230 volts with a maximum allowed deviation of 6% (therefore the maximum consumer volt drop allowed is 13.8 volts). For the purposes of simplicity, non-standard loads were not used and the costs of losses were not considered, although the software tool is capable of handling both of these factors.

VI. CONCLUSIONS

A tool that performs an optimisation on the design of radial LV distribution networks has been presented. It operates on the network's topology, by changing the positions of junctions in the network. The goal of the optimisation is to minimise the associated network cost, while obeying the maximum volt drop constraint. The network cost is the sum of the cost of cables and the cost of power losses in cables, but it can be easily modified to consider other costs, without changing the algorithm.

There have been previous investigations into the development of techniques that optimise LV networks by searching for optimal cable type configurations. The tool presented in this paper uses a different approach, to address the situation in which the designer's choice of cable types is not variable or only slightly variable.

The tool has been applied to existing network designs and achieved a 7% saving in the cost of cables for a specific case study. If the criteria for suitability of application to a particular type of system is the ease with which alternative junction locations can be found, then it is deduced that the tool is most suitable for application to distribution systems in which both service and distributor cable segments are underground (completely underground systems).

such that a cost function is either maximised or minimised. The variables can either be continuous or integral and the problem is usually subject to a set of constraints that, in common with the cost function, can be formulated in terms of these variables.

The Branch and Bound Algorithm entails enumerating the complete set of permutations on all possible values for the integral variables, while performing a standard linear programming optimisation on the continuous variables for each permutation. To "enumerate a permutation" means to evaluate the implications that the permutation has on the cost function and the set of constraints. To "explicitly enumerate a permutation" entails evaluation of the cost function and constraints associated with that permutation. To "implicitly enumerate a permutation" entails inspection of certain criteria that reveal that it is not necessary to explicitly enumerate the configuration due to the implications of these criteria. If a large majority of the permutations are implicitly enumerated by showing that they cannot yield an improved solution over the presently known best solution, then they would not have to be explicitly enumerated, thus reducing the total processing time.

The hierarchy of permutations for the integral variables could be represented by a tree structure, in which each *branch* of the tree corresponds to a particular value for a particular variable. Each *node* in the tree represents a possible permutation. When a group of permutations is implicitly enumerated, the branches of the tree corresponding to these permutations are removed from the tree (*pruned*) in an operation that is called a *bound*. Therefore all nodes in that branch are removed from the search. In the problem of junction position optimisation, each junction is represented by an integral variable, whose possible values correspond to each position. The criteria for implicit enumeration, which must be chosen according to the problem at hand, are discussed later.

The process used in AJPO is made up of three parts, namely pre-processing, pre-checking and searching.

Pre-processing entails preparation of data that is required by the search. This is done by performing calculations using the input data that AJPO receives from the user. Values of cost, current and volt drop associated with each cable segment are calculated.

The cost of each junction group corresponding to each of its possible junction positions is calculated, by summing the associated IDCS, ODCS and SCS cable segment costs.

The value of current in each cable segment is calculated with consideration for all connected consumers and diversity of their loads. A check is made to determine if the value of current in any of the cable segments exceeds the cable type's maximum rated value, in which case this is made known to the user.

Volt drops are calculated in each cable segment using the values for current and with compensation for possible unbalance between phases in 3 phase cable segments. Unbalance between phases occurs when the loads on each of the phases are not all equal.

Each junction group has certain significant volt drops associated with each position. These include: the volt drop of each ODCS added to the volt drop in the IDCS, and the volt drop in the IDCS added to the largest SCS volt drop. Apart from testing the satisfaction of the maximum volt drop constraint, these volt drop values are also used to enable AJPO to incrementally change a junction's position in an order that results in an incremental increase in maximum consumer volt drop.

The next task of this stage is to ascertain which junction positions are redundant. A redundant position is one that has an associated cost that is greater than another position's associated cost, while simultaneously having an equal or larger maximum consumer volt drop. Only the remaining positions are considered by AJPO, and therefore the incremental change in a junction's position that results in an incremental increase in maximum consumer volt drop, will also result in an incremental decrease in cost.

Pre-checking involves two simple tests that could result in finding a solution without the need to perform a search. The first test entails placing all junctions in their cheapest positions, and checking the maximum volt drop constraint. If no violation occurs, then this configuration is the cheapest feasible solution. The other test entails placing all junctions in their positions that result in lowest maximum consumer volt drops. If this configuration causes a maximum volt drop constraint violation, then there will be one under all configurations. Therefore, if an attempt to find a solution was made, it would not be found.

The search is based on an adaptation of the branch and bound search method. It entails searching for the combination of junction positions (permutation), that results in minimum total network cost and satisfies the maximum volt drop constraint. Each combination must be investigated to find the solution, either by implicit or explicit enumeration. It was mentioned earlier that the root of a distributor sub-network is the cable segment at the supply side of the radial network. The junction on the load side of this cable segment is termed the *root junction* (RJ) of that sub-network. Each of the junction groups that are directly connected to a RJ via its ODCSs (the *children* of this RJ) are the RJ's of their own sub-network. A junction group's IDCS is connected to an ODCS of its *parent*'s junction group.

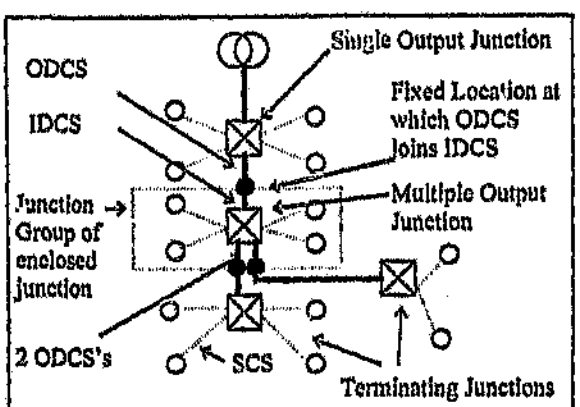


Fig. 2. Illustration of Junction Groups

The geographic positions of junctions within junction groups influence the routes and therefore lengths of cable segments within a network. The lengths of cable segments determine the network cost and dictate the possibility of a violation of the maximum volt drop constraint. This is because the cost and volt drop associated with a cable segment are both directly proportional to its length.

If the position of a particular junction is changed by moving it to another location, then the routes of all of its connected cable segments would have to be adjusted according to the change. Therefore the network's cost and maximum consumer volt drop would change. The network's cost is the value of the cost function calculated for the network in its current configuration. Of the two network configurations that correspond to each junction location, the configuration with the lower network cost would be preferable. However, if it causes a violation of the maximum volt drop constraint, then the more expensive network design would have to be chosen.

Consider the situation in which a set of alternative locations is specified for each junction. If there are n junctions and each junction has p_j possible positions (for $1 \leq j \leq n$), then the network can have p_t different possible configurations, where:

$$p_t = \prod_{j=1}^{n} p_j \qquad (4)$$

These different possible configurations each represent one permutation. For each permutation there is an associated network cost and maximum consumer volt drop. Of all the permutations, the configuration with the lowest associated cost which does not cause a violation of the maximum volt drop constraint should be chosen.

AJPO has been developed to find this permutation. Therefore its purpose is to find the optimal positions of a finite number of junctions, each having a finite number of possible locations. The problem is subject to the maximum volt drop constraint and the optimal solution lies at a configuration where the cost function is a minimum. AJPO is supplied with the necessary input information, which includes:

- The lengths of all cable segments in all junction groups corresponding to all possible junction positions.
- Cable type information for each cable segment.
- Information required to calculate the values of current and volt drop corresponding to cable segments.
- The value of the largest allowed maximum consumer volt drop for adherence to the maximum volt drop constraint.

The mechanism that compiles the set of alternative locations for each junction is independent of AJPO. It is the user's responsibility to find these positions. They could be specified interactively or found by the suitable application of artificial intelligen. , such as an expert system. They could also possibly be extracted form a Geographic Information System, if AJPO was integrated into such a system. The choice of junction positions should consider pragmatic issues, and therefore they are often restricted to certain specific locations. The mechanism that calculates the lengths of cable segments is also independent of AJPO. Cable routes could also be specified interactively or found by an automatic routing tool.

The problem is analogous to one addressed by Nicolson[5] in which cable type selection software was developed. This software finds cable types for all cable segments in a distributor sub-network, such that the solution has a minimum associated cost and does not violate the maximum volt drop constraint. In the problem of finding optimal junction positions, the difference is that instead of searching for cable types for each cable segment out of a finite set of available cable types, the search is for junction positions for each junction out of a finite set of available locations. The goal of finding a configuration having a minimum network cost and the adherence to the maximum volt drop constraint are common to both problems. An adaptation of the Branch and Bound method was chosen for cable type selection[5], because of its property of being able to guarantee an optimal solution within its allotted search space.

The Branch and Bound Algorithm is a technique used for solving mixed integer programming (MIP) problems [12]. A mixed integer program is a mathematical model in which the intention is to find values for a set of variables

The goal of the optimisation is to find a value for $\underline{x}$ that results in a minimum or maximum value of C, i.e. to find $\underline{x}$ such that (for minimum):

$$C = \min f(\underline{x}) \quad (2)$$

In the problem of electrical distribution design, the variable that is to be optimised is the overall cost of the network, for which a minimum value is to be found. The cost of a network is made up of various components, including:

- The cost of designing the network is typically proportional to the amount of time that a designer is required to spend on developing the design. The use of CAD tools should reduce this time considerably and therefore inherently reduce the cost of designing the network. The processing requirements of a particular CAD tool should be minimised so that the required time for its use is minimised.
- The cost of network elements, is the sum of the costs of transformers, cable segments and junctions and other network elements.
- The cost of installation of elements
- The cost of maintenance to the network
- The cost of losses associated with using the LV network, is made up of the cost of losses in the LV system components, such as the cost of power losses within cable segments over the system's period of use. The cost of losses in a particular cable segment is proportional to the power dissipated in that cable segment, which is proportional to the square of the average current. The proportionality factor depends on the load factor, the cost of electricity and the time over which the system is to be used.
- The costs of interruptions and failure to provide the required quality of supply, are not considered in the calculations because they are very hard to quantify. The required quality of supply is dealt with by applying the maximum volt drop constraint as a design constraint.

The total cost is the sum of all the above mentioned factors. However, due to an inability to accurately quantify some of the factors, they have not been included for further consideration. The factors that are omitted include: the cost of designing the network; the cost of installation of elements; the cost of maintenance to the network; and the costs of interruptions and failure to provide the required quality of supply. The cost of network elements is only made up of the cost of cable segments, because AJPO does not change the number and type of junctions and transformers.

Therefore, the cost function that is to be minimised is the sum of the costs of all the cable segments used in the network and the sum of the life-time costs. If there are q cable segments in a particular network, then its associated cost is:

$$C = \sum_{j=1}^{q} L_j \cdot cpul_j + K_j * \left(I_j\right)^2 \quad (3)$$

where:

L_j is the length of cable segment j

$cpul_j$ is the cost per unit length of cable segment j

I_j is the average current flowing in cable segment j

K_j is the proportionality factor for losses in cable segment j

The optimisation algorithm is subject to the *maximum volt drop constraint*, which stipulates that the voltage drop between the transformer and all consumers in the network must not exceed a certain amount. This means that the network's maximum consumer volt drop must not exceed this amount.

IV. DESCRIPTION OF ALGORITHM THAT FINDS OPTIMAL JUNCTION POSITIONS

This section provides a description of AJPO. AJPO operates on a single distributor sub-network (illustrated in figure 1). The network is decomposed into a set of entities called *junction groups*, each corresponding to one of the junctions in the network. A junction group includes all consumers that are directly connected to that junction; their corresponding service cable segments; a portion of the distributor cable segment that supplies the junction; and portions of the distributor cable segments that are supplied by this junction. This concept is illustrated in figure 2.

Each junction is connected to the load side of a single distributor cable segment. A portion of this cable segment is referred to as the junction's *input distributor cable segment (IDCS)*. Each junction is normally directly connected to a number of consumers by the junction's *service cable segments (SCS)*. Each junction is connected to the supply side of zero or more distributor cable segments that connect to other junctions. Portions of these cable segments are called *output distributor cable segments (ODCS)*. Junctions that have more than one ODCS are called *multiple output junctions*. Junctions with only one ODCS are called *single output junctions*, and junctions with none are called *terminating junctions (TJ)*.

The combination of an ODCS and an IDCS collectively make up a distributor cable segment. They connect at a fixed location. An example of how one junction's ODCS is connected to another junction's IDCS at a fixed location is shown in figure 3 by the cable segments pointed to by the labels in the top left corner of the figure. In reality this connection is only one cable segment, but for the purposes of this model they will be considered as separate cable segments.

positions of LV junctions have on the cost of a distribution network, by searching for a configuration of junction positions that results in a minimum associated network cost. The tool is called Automated Junction Position Optimisation (AJPO). The maximum volt drop and maximum current constraints are considered. The search algorithm that is used by AJPO is an adaptation of the branch and bound technique.

The development of this tool is part of a larger ongoing research effort that is being carried out at the University of the Witwatersrand[10]. This research has focused on the development of software tools that are intended to automate and facilitate various aspects of the electrical distribution network design process.

II. BACKGROUND TO DISTRIBUTION DESIGN

To appreciate the functionality of a tool that aids in the distribution design process, it is necessary to understand the typical structure of a distribution network and the process of designing a distribution network.

In an electrical distribution network there are normally a large number of consumers that are connected to a source of electricity by a particular hierarchy of facilities. The hierarchy includes a medium voltage (MV) portion and an LV portion. The connection between these two portions occurs at an MV/LV transformer. Figure 1 illustrates the LV portion, which usually has the structure of a radial network.

In a radial LV network there is normally a *transformer* that is directly connected to a number of *junctions* by *distributor cable segments*. Current flows from the transformer to these junctions, therefore the transformer end of the distributor cable segment is at the *supply* side and the other end is the *load* side. Each junction is connected to the supply side of zero or more distributor cable segments, which are connected to other junctions on their load sides. *Service cable segments* are connected between junctions or transformers (at the supply side) and *consumers* (at the load side). The network is purely radial and therefore there are no interconnected links between elements. The current flowing through a cable segment in the network causes a volt drop to occur across the cable segment. Therefore the voltage level is lower at a consumer than at a transformer, by an amount that is the sum of the volt drops across the individual cable segments along the route between the transformer and the consumer. The largest value for this total volt drop amongst all consumers in a network is called the *maximum consumer volt drop*.

The term '*distributor sub-network*' is used to refer to a portion of a radial network. The *root* of a distributor sub-network is a particular distributor cable segment. The corresponding distributor sub-network is made up of that cable segment as well as all junctions, cable segments and consumers that make up the network that radiates from the root (from the supply to the load). For example, the two distributor cable segments that are connected to the transformer in figure 1 are the roots of the two corresponding distributor sub-networks. Any distributor cable segment in the network is the route of its distributor sub-network.

When a network is required for a particular area, the designer's first step is usually to obtain or create a map of that area. An adequately detailed map will show the locations of consumers as well as the locations of surrounding sources of electricity. The designer's task is to design an electrical distribution network that connects all the consumers to a source of electricity. It is usually the designer's goal to find a network configuration that has a minimum a- -iated cost while still satisfying the applicable constraints and requirements. The process of creating the network design is influenced by a mixture of governing factors, including: network requirements; budget; available design tools; and the technique which the designer is accustomed to.

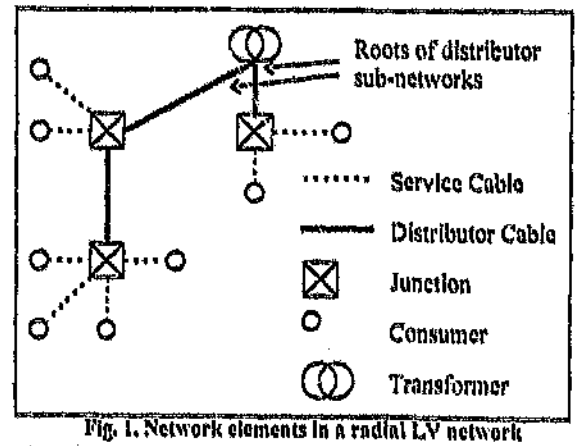


Fig. 1. Network elements in a radial LV network

III. BACKGROUND TO OPTIMISATION

According to Dixon [11], in any optimisation problem, there is some function which is to be optimised, i.e. the entity that is to obtain its 'least' or 'greatest' value at the solution of the problem. This function depends on one or more independent system variables and is called the cost function C. These independent variables, which can be formed into a vector $\underline{x}$, define the space in which the problem is posed. In the 'static' problem the cost C is a function of $\underline{x}$, i.e.:

$$C = f(\underline{x}) \qquad (1)$$

Junction Position Optimisation in Radial Low Voltage Electrical Distribution Network Design

J Apostolellis
University of the Witwatersrand, Johannesburg, South Africa

Abstract-The use of computer-based tools in the design of electrical distribution networks results in the cost effective production and implementation of designs. This is because they help automate time-consuming routine tasks and enable the optimisation of certain aspects of a network design. A tool that performs an optimisation on the LV portion of a distribution network is discussed. It uses the influence that a network's topology has on its associated cost, by applying a search that modifies the topology to result in a minimum cost. The modification of the network's topology is achieved through changing junction positions to alternative locations. It operates on an entire radial LV network, and therefore considers the coupling between junctions in the same network. The search algorithm is based on an adaptation of the branch and bound method. Test results revealed a 7% saving in the cost of cables for a specific case study.

I. INTRODUCTION

A large percentage of households in South Africa and other developing countries are currently without electricity. Major efforts are being carried out to address the need for improved electrification. One of the tasks in bringing electricity to a community is the design of the distribution network that connects consumers to a source of electricity. The production of good network designs is of great importance in developing countries. This is because large amounts of money could be saved if good network designs are used for the implementation of electrical distribution networks. A good network design is one whose associated cost of production and implementation is as low as possible, while all imposed requirements and constraints are satisfied. Therefore they are termed 'optimal network designs'.

There have been numerous investigations into the use of computers to aid the process of producing distribution network designs. The late 1950's marked the beginning of work in this field, with the publication by Grimsdale et al [1] in which a computer based system is discussed that automatically finds substation locations and feeder routes for the design of a distribution network for a housing-estate.

Much has been published on the modelling of the primary (MV) portions of distribution networks, for the purposes of optimal design and plan production. A recent publication by Glamocanin et al [2] provides a brief discussion about such previous work. There have also been various investigations into the modelling and implementation of automated distribution systems, which perform the task of restoring and restructuring the network under overloading or fault conditions. These systems usually operate on non-radial MV distribution networks.

Little has been published about the optimal design of the secondary (LV) portions of distribution networks. There have been various investigations into the development of techniques that optimise LV network designs by searching for optimal cable type configurations [3-5]. None of these techniques optimise the design by exploring the effects of changes to the network's topography. The approach of minimising cost by optimal cable type selection is of limited use in the situation in which there are a small number of available cable types for a design.

The development and use of a rule-based expert system for the design of radial secondary distribution systems is discussed in [6]. A solution is found via the application of the rule-base. In a later publication by the same authors[7], they state that an expert system approach has several advantages over algorithm-oriented optimisation techniques. One of these advantages is that conventional optimisation techniques can be included as callable routines that are invoked by the expert system. However no particular optimisation routines have been mentioned.

Sunde et al [8,9] have developed an Intelligent Decision Support System (IDSS) for Automated Electrical Plat Design (AEPD). This system produces electrical distribution network designs via the application of artificial intelligence through which heuristic strategies and rules of thumb are automatically applied. It mentions that optimisation routines can still be used to solve some segments within the design process, but there is no indication that any optimisation routines have actually been incorporated into the system.

The summary of published literature presented above demonstrates that there is a need for specific CAD-based tools that are to be used in the automation of the overall network design process. Work has been published about tools that optimise LV network designs without changing the network's topology[3-5]. This paper discusses the development of a tool that performs an optimisation on radial LV distribution networks by changing the network's topology. It takes advantage of the influence that the

The Configuration Management Plan (JA 006) provides a specification of artefact management and storage. In particular, it explains the artefact naming system that has been adopted, and points out that all artefacts are categorised as one of management products, technical products or quality assurance products.

The Literature Survey Report (JA 115) discusses the findings of the literature survey that was conducted as an essential part of this project. The survey was focused on previously published work in the field of CAD-based automation and optimisation tools applied to the design of electrical distribution networks.

The Algorithm Description (JA 116) provides a detailed description of the algorithm that is implemented in the software tool, which is called Automated Junction Position Optimisation (AJPO). It discusses the background to distribution design and an initial prototype that was developed prior to the actual specification of the algorithm that is used in AJPO.

The High Level Software Design (JA 117) provides a high level view of the AJPO software. It discusses the approach to software development that was applied. It also serves as a repository for information that was produced during the early stages of software development.

The Software Test Specification (JA 118) specifies the test procedures that were to be performed to quantify the accuracy and functionality of the software. It also provides an evaluation of the software by discussing the suitability of applying AJPO to various types of distribution networks.

The Low Level Software Design (JA 124) provides detailed descriptions of all classes in the AJPO software. It does so by documenting all member functions and data members for each class.

The Software Test Result Report (JA 125) discusses the results of the tests that are specified in the Software Test Specification (JA 118). It reveals that the accuracy and functionality of the software is as required. It also shows that savings of 7% in cable costs were achieved for a particular case study with the application of the AJPO tool.

The Requirements Verification and Validation document (JA 130) provides a cross reference to all requirements identified in documents such as the Product Description, Product Functional Specification, Algorithm Description, High Level Software Design, Software Test Specification and Software Test Result Report. It illustrates how the functional requirements were transformed between stages in the software development process, from conception to realisation.

FOREWORD

The format for this Masters Dissertation consists of a short body (in the form of a technical paper) and multiple appendices. It therefore differs from the standard dissertation format and is said to be of the 'Paper Model' format. The technical paper presents a brief view of the core of the project, while the appendices contain the substance of the work that was performed in this project. All documents that are products of this project are named with 'JA' as a prefix and a specific number as a suffix.

The project has been conducted in an ISO 9001 accredited software development environment, called the Software Engineering Applications Laboratory (SEAL). It is standard SEAL practice to associate each project with a unique name, and a corresponding acronym. This project is called Network Optimisation in Distribution Design (NODD). It deals with the design of electrical distribution networks, by investigating the application of optimisation to network designs. This is achieved through the development of a CAD-based tool called Automated Junction Position Optimisation (AJPO).

The paper is titled "Junction Position Optimisation in Radial Low Voltage Electrical Distribution Network Design". It discusses how the software tool that was developed in this project fits into the overall field of CAD-based distribution design tools. The algorithm that is used by this tool is thoroughly described and a case study is presented to illustrate how the tool is applied to a distribution network. It therefore provides a concise view of both the research and the application of the software tool that was developed in this project.

The Conclusions and Recommendations for Future Work (JA 135) provides a summary of the findings in the project. It also discusses the possibility of related work for the future.

The Project Bibliography (JA 140) provides a list of all published papers and reports that were encountered and consequently referenced in the project documentation.

The appendices in this dissertation, which are categorised into one of two appendix groups, are as follows:

The Master Document List (JA 001) serves as a register of all documents created in the project. It is a key document in the project, because it facilitates tracking of all other documents, as well as provides a view of the entire project from a documentation perspective. It serves as a centre of control for the project, by storing information about the locations and revisions of all documents.

TABLE OF CONTENTS

Page Number/
Document Number

Declaration ii

Abstract iii

Table of Contents iv

Foreword v

MSc Paper - J. Apostolellis JA121

Conclusions and Recommendations for Future Work JA135

Project Bibliography JA140

Appendix Group I: Management Products

Master Document List JA001

Configuration Management Plan JA006

Appendix Group II: Technical Products

Literature Survey Report JA115

Algorithm Description JA116

High Level Software Design JA117

Software Test Specification JA118

Low Level Software Design JA124

Software Test Result Report JA125

Requirements Verification and Validation JA130

ABSTRACT

The use of computer-based tools in the design of electrical distribution networks results in the cost effective production and implementation of designs. This is because they help automate time-consuming routine tasks and enable the optimisation of certain aspects of a network design.

The production of a tool that performs an optimisation on the low voltage (LV) portion of a distribution network is discussed. It uses the influence that a network's topology has on its associated cost, by applying a search that modifies the topology to result in a minimum cost. The modification of the network's topology is achieved through changing junction positions to alternative locations. It operates on an entire radial LV network, and therefore considers the coupling between junctions in the same network. The user of the tool must specify the alternative locations for each junction in the network, as well as information about the network, such as cable lengths and cable types.

The search algorithm is based on an adaptation of the branch and bound method, which is a reduced search algorithm. It has been chosen as opposed to an exhaustive search algorithm to reduce computational requirements.

The tool is most useful for optimising completely underground distribution systems. Test results revealed a 7% saving in the cost of cables for a specific case study.

The software has been developed with the application of object-oriented analysis and design within an ISO 9001 accredited software development environment.

DECLARATION

I declare that this dissertation is my own work. It is being submitted for the degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University.

(Signature of candidate)

25th day of February 1997

THE PRODUCTION OF SOFTWARE THAT AIDS IN THE DESIGN OF LOW VOLTAGE DISTRIBUTION NETWORKS BY OPTIMISING THE LOCATIONS OF JUNCTIONS

Justin Apostolellis

A dissertation submitted to the Faculty of Engineering, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, 1996

Change History

Configuration Control

Project:	NODD
Title:	Master Document List
Doc. Reference:	C:\1995_09\MP\JA001.100
Created by:	Justin Apostolellis
Creation Date:	19 April, 1995

Document History

Version	Date	Status	Who	Saved as:
0.01	96/03/01	Draft	JA	C:\1995_09\MP\JA001.001
1.00	96/03/04	Approved	JA	C:\1995_09\MP\JA001.100
1.00	96/08/26	Approved	JA	C:\1995_09\MP\JA001.100
1.00	96/08/28	Approved	JA	C:\1995_09\MP\JA001.100
1.00	96/09/16	Approved	JA	C:\1995_09\MP\JA001.100
1.00	96/09/17	Approved	JA	C:\1995_09\MP\JA001.100
1.00	96/09/19	Approved	JA	C:\1995_09\MP\JA001.100
1.00	96/10/16	Approved	JA	C:\1995_09\MP\JA001.100
1.00	96/10/17	Approved	JA	C:\1995_09\MP\JA001.100
1.00	96/10/26	Approved	JA	C:\1995_09\MP\JA001.100
1.00	96/10/31	Approved	JA	C:\1995_09\MP\JA001.100
1.00	96/11/07	Approved	JA	C:\1995_09\MP\JA001.100
1.00	96/11/13	Approved	JA	C:\1995_09\MP\JA001.100
1.00	96/12/02	Approved	JA	C:\1995_09\MP\JA001.100
1.00	96/12/18	Approved	JA	C:\1995_09\MP\JA001.100

Revision History

Version	Date	Changes
0.01	95/04/19	Creation of document from template : QST00110.101
0.01	95/06/12	Correction of date formats and JA100 in section 2

Table of Contents

Table of Contents 2

Change History 3

Configuration Control 3

Document History 3

Revision History 3

Management Authorisation 5

Change Forecast 5

1 Scope 6

1.1 Introduction 6

1.2 Purpose 6

1.3 Applicability 6

1.4 Definitions 6

1.5 Notes 6

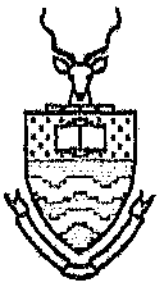
1.6 Audience 6

1.7 Applicable Documents 7

1.8 Assumptions 7

1.9 Requirements Traceability 7

2 Master Document List 8



NODD

Master Document List

Management Product

Version 1.00

Document Status: Approved

[51]Dixon L "Non-Linear Optimization" Bell and Bain, 1972.

[52]Kirkpatrick S Gelatt C D J, Vecchi M P "Optimisation by Simulated Annealing" Science, vol. 230, pp. 671-680, May 1983.

[53]Holland J "Adaptation in natural and artificial systems" University of Michigan Press, 1975.

[54]Salkin, H, Mathur, K and Hass, R "Foundations of Integer Programming", North-Holland, 1989.

[55]Bellman R E "Dynamic Programming" Princeton University Press, 1957.

[56]Booch G "Object-Oriented Analysis and Design with applications" Second Edition, The Benjamin/Cummings Publishing Company, Inc., 1994.

[57]Booch G "Object-Oriented Development" IEEE Transactions on Software Engineering, Vol. SE-12, No. 2, February 1986.

[58]Stroustrup B "The C++ Programming Language" Second Edition, AT&T Bell Laboratories, Murray Hill, New Jersey, 1991.

[59]Rumbaugh J, Blaha M, Premerlani W, Eddy F, Lorensen W "Object-Oriented Modelling and Design" Prentice-Hall Inc., 1991.

[60]Nitch D "A serial and parallel design of NEC2 to demonstrate the advantages of the object-oriented paradigm in comparison with the procedural paradigm" Ph. D. Thesis, Electrical Engineering, University of the Witwatersrand, 1992.

[61]Boehm B "A Spiral Model of Software Development and Enhancement" Software Engineering Notes Vol. 11 (4), August 1986.

[38]Sumic Z, Venkata S S, Pistorese T, "Automated underground residential distribution design, Part 2 : Prototype implementation and results", IEEE Transactions on Power Delivery, Vol. 8, No. 2, April 1993.

[39]Yeh E C, Sumic Z, Venkata S S, "APR: A Geographic Information System Based Primary Router for Underground Residential Distribution Design" IEEE Transactions on Power Delivery, Vol. 10, No. 1, February 1995.

[40]Broadwater R, Thompson J, Ellis M, Ng H, Singh N, Loyd D "Application Programmer Interface for the EPRI Distribution Engineering Workstation" IEEE Transactions on Power Systems, Vol. 10, No. 1, February 1995.

[41]Meyer A, Dwolatzky B "The effective computer generation of designs for township electrical distribution", Proceedings of the IEEE Power Engineering Society, Transmission and Distribution Society, Dallas, Texas, September, 1991.

[42]Trudeau D, Hoffman R, Seymour M A "Integrating AM/FM Maps with Distribution SCADA" IEEE Transactions on Power Delivery, Vol. 5, No. 2, April 1990.

[43]Carr J, McCall L V "Divergent Evolution and Resulting Characteristics among the World's Distribution Systems" IEEE Transactions on Power Delivery, Vol. 7, No. 3, July 1992.

[44]Yu D C, Flinn, D G, Krieger R A "Facilitating Engineering Analysis Via a Graphical Database" IEEE Transactions on Power Systems, Vol. 10, No. 1, February 1995.

[45]Badenhorst C, Dingley C E, Ferguson I "An Investigation into the use of Intermediate Voltage for Rural Electrification" South African Universities Power Engineering Conference, Pretoria, January 1995.

[46]Wyatt G, Dingley C "A Financial Comparison of Three-Phase versus Single-Phase Rural Electricity Distribution" South African Universities Power Engineering Conference, Pretoria, January 1995.

[47]Whitaker, T "Personal Communications", 1996.

[48]ESKOM "Electrification Standard", Revision 2, April 1993.

[49]Dwolatzky B and Meyer A S "The development of an integrated set of software tools for use in the design of an electrical reticulation network" South African Universities Power Engineering Conference (SAUPEC) 95, Pretoria, January 1995.

[50]West N A, Dwolatzky B and Meyer A S "Software which automatically Routes Cables in a Power Reticulation Design System" The Transactions of the South African Institute of Electrical Engineers" September 1994.

[24]Shirmohammadi D, Wayne H 'Reconfiguration of electric distribution networks for resistive line losses reduction' IEEE Transactions on Power Delivery, Vol. 4, No. 2, April 1989.

[25]Liu C C, Lee J L, Vu K 'Loss Minimisation of Distribution Feeders: Optimality and Algorithms' IEEE Transactions on Power Delivery, Vol. 4, No. 2, April 1989.

[26]Goswami S K, Basu S K "A new algorithm for the reconfiguration of distribution feeders for loss minimisation" IEEE Transactions on Power Delivery, Vol. 7, No. 3, July 1992.

[27]Carson M J, Cornfield G 'Design of low-voltage distribution networks' Proceedings of the IEE, Vol. 120, No. 5, May 1973.

[28]Hindi K S, Brameller M I, Gas E 'Design of low-voltage distribution networks: a mathematical programming method' Proceedings of the IEE, Vol. 124, No. 1, January 1977.

[29]Chia C S 'Optimised Distribution Design' Ph. D. thesis, UMIST. 1.

[30]Snelson B A, Carson M J 'Logical design of branched LV distributors" Proceedings of the IEE, Vol. 117, No. 2, February 1970.

[31]Walkden F W "Design of low-voltage distributors" Proceedings of the IEE, Vol. 129, Pt. C, No. 3, May 1982.

[32]Walkden F W "Optimising the design of low-voltage cable networks" Proceedings of the IEE, Vol. 133, No. 1, January 1986.

[33]Nicolson, J M "Object oriented design of cable selection software for low voltage networks", Master's Thesis, University of the Witwatersrand, Johannesburg, April 1993.

[34]Whitaker T, Design Engineer at Johannesburg Electricity Department "Personal Communications", 1996.

[35]Shao J, Rao N D, Zhang Y "An Expert System for Secondary Distribution Design" IEEE Transactions on Power Delivery, Vol. 6, No. 4, October 1991.

[36]Rao N D, Zhang Y "An Intelligent Front End for Secondary Power Distribution System Design" IEEE Transactions on Power Delivery, Vol. 7, No. 2, April 1992.

[37]Sumic Z, Venkata S S, Pistorese T, "Automated underground residential distribution design, Part 1 : Conceptual design", IEEE Transactions on Power Delivery, Vol. 8, No. 2, April 1993.

[12]Boardman J T, Meckiff C C 'A Branch and Bound Formulation to an Electricity Planning Problem' IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, No. 8 August 1985.

[13]Youssef H K, Abu-El-Magd M A 'Novel Optimisation Model for Loong Range Distribution Planning' IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, No. 11 November 1985.

[14]Gonen T, Ignacio J R R 'Optimal Multi-Stage Planning of Power Distribution Systems' IEEE Transactions on Power Delivery, Vol. PWRD-2, No. 2, April 1987.

[15]Ponnavaiko M, Prakasa Rao K S, Venkata S S 'Distribution System Planning through a Quadratic Mixed Integer Programming Approach' IEEE Transactions on Power Delivery, Vol. PWRD-2, No. 4, October 1987.

[16]Youssef H K, Hackam R 'Dynamic Solution of Distribution Planning in Intermediate Time Range' IEEE Transactions on Power Delivery, Vol. 3, No. 1, January 1988.

[17]Glamocanin V 'Optimal Loss Reduction of Distribution Networks' IEEE Transactions on Power Systems, Vol. 5, No. 3, August 1990.

[18]Ben-Dov E, Harley R G, Seymore W J 'Design of an optimal reticulation system for a residential area' IEEE Transactions on Power Systems, Vol. PWRS-2, No. 1, February 1987.

[19]Willis H L, Powell R W, Vismor T D 'A method of automatically assessing load transfer costs in substation optimisation studies' IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, No.10 October 1985.

[20]Chen J L, Hsu Y Y "An Expert System for Load Allocation in Distribution Expansion Planning" IEEE Transactions on Power Delivery, Vol. 4, No. 3, July 1989.

[21]Hasselfield C W, Wilson P, Penner L, Lau M, Gole A M "An Automated Method for Least Cost Distribution Planning" IEEE Transactions on Power Delivery, Vol. 5, No. 2, April 1990.

[22]Jones A I, Smith B E, Ward D J "Considerations for Higher Voltage Distribution" IEEE Transactions on Power Delivery, Vol. 7, No. 2, April 1992.

[23]Morelato A L, Monticelli A 'Heuristic Search Approach to Distribution System Restoration' IEEE Transactions on Power Delivery, Vol. 4, No. 4, October 1989.

1 PROJECT BIBLIOGRAPHY

[1] Grimsdale R L, Sinclare P H 'The design of housing-estate distribution systems using a digital computer', Proceedings of the IEE, 1960, 107A, pp 295-305.

[2] Knight U G W 'The Logical Design of Electrical Networks Using Linear Programming Methods' , Proceedings of the IEE, 1960, 107A, pp 306-314.

[3] Boardman J T, Hogg B W 'Computer method for design of electricity-supply networks' Proceedings of the IEE, Vol. 119, No. 7, July 1972.

[4] Crawford D M, Holt S B 'A mathematical optimisation technique for locating and sizing distribution substations, and deriving their optimal service areas' IEEE Transactions on Power Apparatus and Systems, Vol. PAS-94, No. 2 March/April 1975.

[5] Wall D L, Thompson G L, Northcote-Green J E D 'An Optimisation Model for Planning Radial Distribution Networks' ,IEEE Transactions on Power Apparatus and Systems, Vol. PAS-98, No. 3 May/June 1979.

[6] Wall D L, Thompson G L 'A Branch and Bound Model for choosing Optimal Substation Locations', IEEE Transactions on Power Apparatus and Systems, Vol. PAS-100, No. 5 May 1981.

[7] Gonen T, Foote B L 'Distribution-system planning using mixed-integer programming' IEE Proceedings, Vol. 128, Pt. C, No. 2, March 1981.

[8] Sun D I, Farris, D R, Cote P J, Shoults R R, Chen M S 'Optimal Distribution Substation and Primary Feeder Planning via the Fixed Charge Network Formulation', IEEE Transactions on Power Apparatus and Systems, Vol. PAS-101, No. 3 March 1982.

[9] Fawzi T H, Ali K F, El-Sobki S M 'Routing Optimisation of Primary Rural Distribution Feeders', IEEE Transactions on Power Apparatus and Systems, Vol. PAS-101, No. 5 May 1982.

[10]Fawzi T H, Ali K F, El-Sobki S M 'A New Planning Model for Distribution Systems', IEEE Transactions on Power Apparatus and Systems, Vol. PAS-102, No. 9 September 1982.

[11]El-Kady M A 'Computer-Aided Planning of Distribution Substation and Primary Feeders', IEEE Transactions on Power Apparatus and Systems, Vol. PAS-103, No. 6 June 1984.

junction's location in overhead systems is that a junction may be configured to cause branching in orthogonal directions, thus its original location has structural integrity that would be compromised if it was changed in any way.

The software development process entailed the production of a series of documents, that supported the implementation of the software. The functional requirements that were identified at the beginning of the process were traced through the various stages to ensure that the final product meets these requirements. Such a process is recommended as part of the ISO 9001 Standard for Software Development.

The Booch [56] method of object-oriented analysis and design for software development was applied as an iterative process within the framework of the software development process mentioned above. This allowed for the creativity required to effectively produce successful software. It can be stated that this micro development process was applied within the framework of the macro development process, which is defined by the production of the series of documents.

The software was also fully tested for the required accuracy and functionality via the production of a test specification and corresponding test results report. The tests and their results revealed that such requirements were satisfied.

There are no recommendations for improving the algorithm that is discussed in this project, but various related recommendations for the development of CAD-based tools that aid in distribution design are made. The AJPO tool would be greatly enhanced if it worked with an automated tool that finds suitable alternative locations for junctions that AJPO would use. This could be done either by the application of a suitable search algorithm that searches in the vicinity of the junction of interest, or by the application of a knowledge based system that applies the findings of the assertion of rules that are aimed at searching for suitable alternative locations. Another enhancement would be the development of a tool that would act in tandem with AJPO, by possibly changing a consumer's connection to the network from one junction to another. This would also probably best be handled by the application of a knowledge based system.

■

1 CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

A tool that performs an optimisation on the design of radial LV distribution networks has been produced in this project. It operates on the network's topology, by changing the positions of junctions in the network. The goal of the optimisation is to minimise the associated network cost, while obeying the voltage drop constraint. The network cost is made up of the cost of cables and the cost of power losses in cables, but it can be easily modified to consider other costs, without changing the algorithm.

There have been previous investigations [30-33] into the development of techniques that optimise LV networks by searching for optimal cable type configurations. The tool presented in this paper uses a different approach, to address the situation in which the designer's choice of cable types is not variable or only slightly variable.

This tool would be of use in systems that are intended to automate the process of distribution design. It could be applied after secondary cable routing has been performed, to modify the design so that its associated cost is reduced.

The tool has been applied to existing network designs and achieved a 7% saving on the cost of cables for a specific case study. If the criteria for suitability of application to a particular type of system is the ease with which alternative junction locations can be found, then it is deduced that the tool is most suitable for application to distribution systems in which both distributor and service cable segments are underground (completely underground systems).

It is less suitable for application to completely overhead systems than application to underground systems, because of the possible pragmatic consequences of selecting alternative locations for junctions. For example, consider the overhead routes of service cable segments between consumers and junctions (which are placed on poles in overhead systems). If a junction's location is changed, then the resultant service cable route may well be too long, or be obstructed by an obstacle that is only significant at this location. This would deem the alternative location as unsuitable. Another problem is that if a junction's location is changed then the separation between poles might exceed the legitimate maximum allowed distance between poles, which is often a very influential factor in overhead designs. This constraint is further complicated when the cable segment shares a route with another servitude such as a telecommunications line.

Systems with overhead distributor cables and underground service cables are also subject to the constraint that the separation between poles may not exceed a certain maximum amount. Another problem with changing a

It is less suitable for application to completely overhead systems than application to underground systems, because of the possible pragmatic consequences of selecting alternative locations for junctions. For example, consider the overhead routes of service cable segments between consumers and junctions (which are placed on poles in overhead systems). If a junction's location is changed then the resultant service cable route may well be too long or be obstructed by an obstacle that is only significant at this location. This would deem the alternative location as unsuitable. Another problem is that if a junction's location is changed then the separation between poles might exceed the legitimate maximum allowed distance between poles, which is often a very influential factor in overhead designs. This constraint is further complicated when the cable segment shares a route with another servitude such as a telecommunications line.

Systems with overhead distributor cables and underground service cables are also subject to the constraint that the separation between poles may not exceed a certain maximum amount. Another problem with changing a junction's location in overhead systems is that a junction may be configured to cause branching in orthogonal directions, thus its original location has structural integrity that could be compromised if it was changed in any way.

This tool would be of use in systems that are intended to automate the process of distribution design. It could be applied after secondary (LV) cable routing has been performed, to modify the design so that its associated cost is reduced.

VII. BIOGRAPHY

Justin Apostolellis received the BSc degree in Electrical Engineering from the University of the Witwatersrand, Johannesburg. He is currently conducting research towards an MSc in Software Engineering for Distribution Design.

VIII. REFERENCES

[1] Grimsdale R L, Sinclare P H "The design of housing-estate distribution systems using a digital computer", Proceedings of the IEE, 1960, 107A.

[2] Glamocanin V, Filipovic V "Open Loop Distribution System Design", IEEE Transactions on Power Delivery, Vol. 8, No. 4, October 1993.

[3] Snelson J K, Carson M J "Logical design of branched LV distributors" Proceedings of the IEE, Vol. 117, No. 2, February 1970.

[4] Walkden F W "Optimising the design of low-voltage cable networks" Proceedings of the IEE, Vol. 133, No. 1, January 1986.

[5] Nicolson, J M "Object oriented design of cable selection software for low voltage networks", Master's Thesis, University of the Witwatersrand, Johannesburg, April 1993.

[6] Shao J, Rao N D, Zhang Y "An Expert System for Secondary Distribution Design" IEEE Transactions on Power Delivery, Vol. 6, No. 4, October 1991.

[7] Rao N D, Zhang Y "An Intelligent Front End for Secondary Power Distribution System Design" IEEE Transactions on Power Delivery, Vol. 7, No. 2, April 1992.

[8] Sumic Z, Venkata S S, Pistorese T, "Automated underground residential distribution design, Part 1 : Conceptual design", IEEE Transactions on Power Delivery, Vol. 8, No. 2, April 1993.

[9] Sumic Z, Venkata S S, Pistorese T, "Automated underground residential distribution design, Part 2 : Prototype implementation and results", IEEE Transactions on Power Delivery, Vol. 8, No. 2, April 1993.

[10] Meyer A, Dwolatzky B "The effective computer generation of designs for township electrical distribution", Proceedings of the IEEE Power Engineering Society, Transmission and Distribution Society, Dallas, Texas, September, 1991.

[11] Dixon, L "Nonlinear Optimisation", Bell and Bain, 1972.

[12] Salkin, H, Mathur, K and Hass, R "Foundations of Integer Programming", North-Holland, 1989.

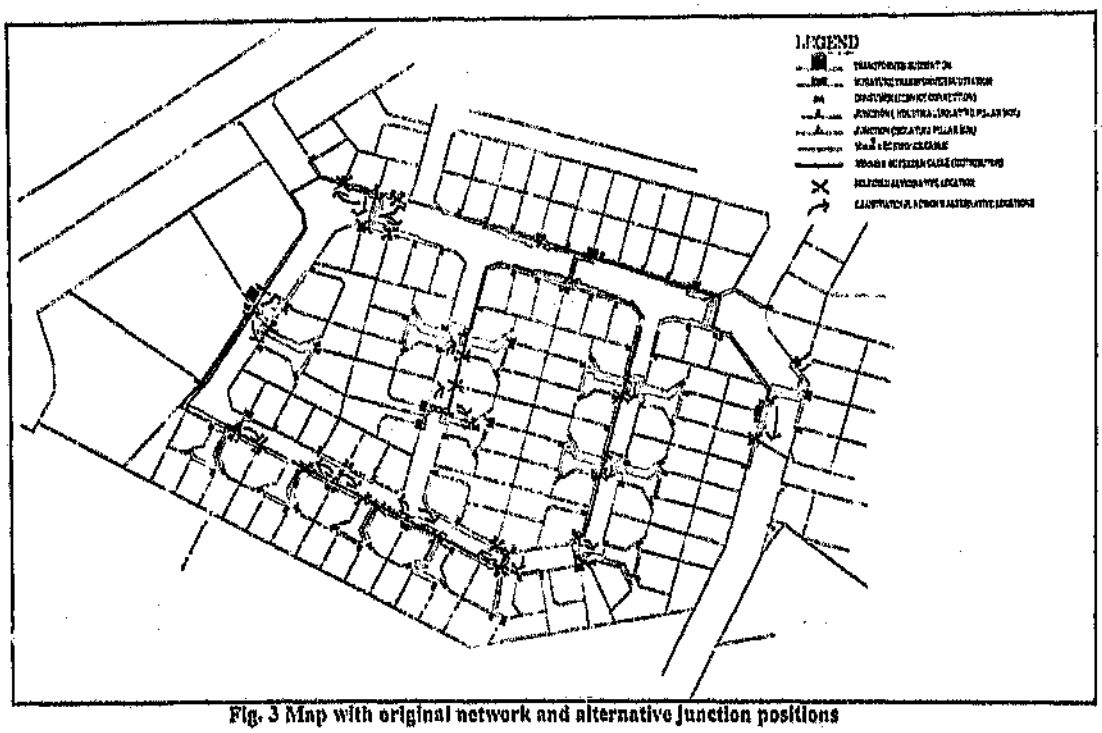


Fig. 3 Map with original network and alternative junction positions

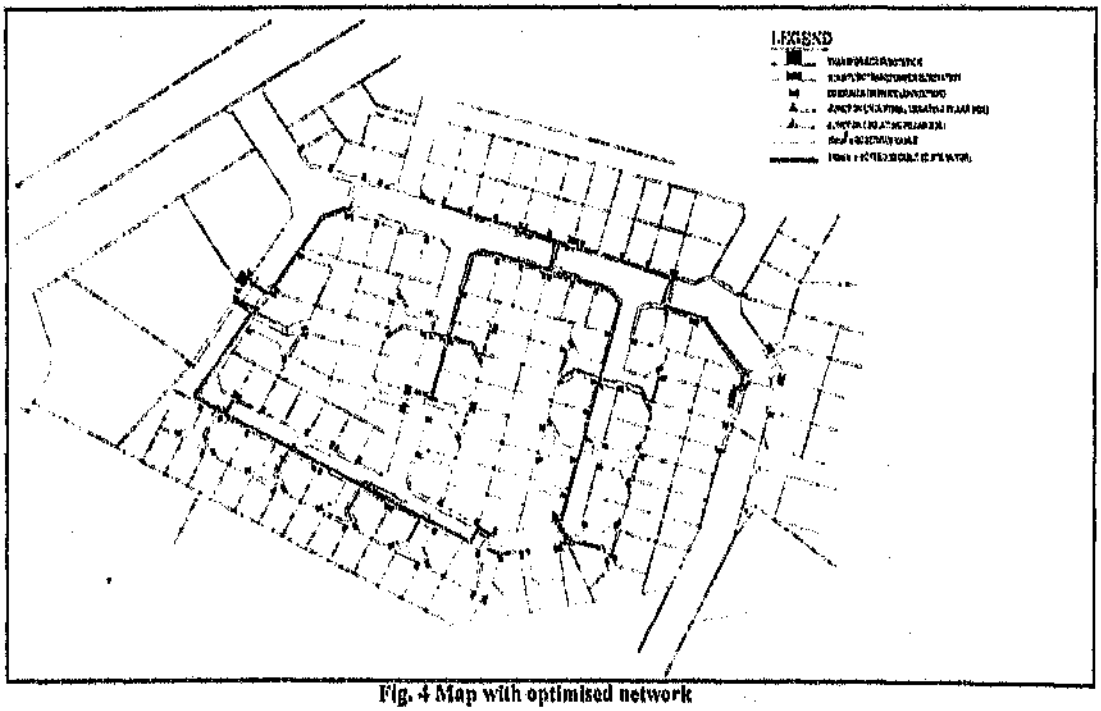


Fig. 4 Map with optimised network

5.1 Electronic Artefacts 12

5.2 Backup procedure on computers in the SEAL Post-graduate Laboratory Facility 12

5.3 Backup procedure on computers outside of the Chamber of Mines Building 12

5.4 Hardcopy Records 12

6 Configuration Status Accounting 13

Table of Contents

Table of Contents 2

Change History 4

Configuration Control 4

Document History 4

Revision History 4

Management Authorisation 4

Change Forecast 4

1 Scope 5

1.1 Introduction 5

1.2 Purpose 5

1.3 Applicability 5

1.4 Limitations 5

1.5 Audience 5

1.6 Applicable Documents 6

1.7 Requirements Traceability 6

2 Documentation Structure 7

3 Artifact Identification 8

3.1 File Naming Convention 8

3.2 Project document sequence numbering 10

4 Artefact Management 11

4.1 Active Machine used for supporting the files 11

4.2 Directory Structure 11

5 Archiving of Artefacts 12



NODD

Configuration Management Plan

Management Product

Version 1.00

Document Status: Approved

Document Name	Document Number	Revision Number	Revision Date	Document Status	Date approved by Board	Minute Reference	File Reference
Project in Progress Audit	JA517	0.01	96/07/30	Approved	96/10/02		C:\1995_09\QA\9609IP07.PAR
Minutes: Project Meeting	JA518	0.01	96/08/28	Approved	96/11/07	JA522	C:\1995_09\QA\960828-1.MIN
Minutes: Project Meeting	JA519	0.01	96/09/17	Approved	96/11/07	JA522	C:\1995_09\QA\960917-1.MIN
Project in Progress Audit	JA520	0.01	96/10/17	Approved	96/12/18		C:\1995_09\QA\9609IP08.PAR
Minutes: Project Meeting	JA521	0.01	96/10/28	Approved	96/11/07	JA522	C:\1995_09\QA\961028-1.MIN
Minutes: Project Meeting	JA522	0.01	96/11/07	Approved	96/12/18	JA524	C:\1995_09\QA\961107-1.MIN
Minutes: Project Meeting	JA523	0.01	96/12/09	Approved	96/12/18	JA524	C:\1995_09\QA\961209-1.MIN
Minutes: Project Meeting	JA524	0.01	96/12/18	Draft			C:\1995_09\QA\961218-1.MIN
Minutes: Project Meeting	JA525	0.01	96/12/24	Draft			C:\1995_09\QA\961224-1.MIN

Document Name	Document Number	Revision Number	Revision Date	Document Status	Date approved by Board	Minute Reference	File Reference
Minutes: JA QMS Project Review Meeting	JA 500	0.01	95/03/23	Approved	95/03/23		C:\1995_09\QA\950323-1.MIN
Project Initiation Audit	JA 501	0.01	95/04/20	Approved	95/04/20		C:\1995_09\QA\9509PI.PAR
Proje[illegible]udit	JA 502	0.01	95/06/15	Approved	95/06/15		C:\1995_09\QA\9509IP01.PAR
M[illegible]oject[illegible]g	JA 503	0.01	95/08/10	Approved	95/08/16		C:\1995_09\QA\950810-1.MIN
Minu[illegible] Project Meeting	JA504	0.01	95/08/16	Approved	95/08/25		C:\1995_09\QA\950816-1.MIN
Project In Progress Au[illegible]	JA505	0.01	95/08/17	Draft			C:\1995_09\QA\9509IP02.PAR
Minutes: Project Meeting	JA50C	0.01	95/08/23	Approved	95/09/22		C:\1995_09\QA\950823-1.MIN
Project In Progress Audit	JA507	0.01	95/10/12	Approved	96/12/06		C:\1995_09\QA\9509IP03.PAR
Project In Progress Audit	JA50R	0.01	95/12/05	Approved	96/12/19		C:\1995_09\QA\9509IP04.PAR
Minutes: Project Meeting	JA509	1.00	95/12/12	Approved	96/03/01	JA510	C:\1995_09\QA\951212-1.MIN
Minutes: Project Meeting	JA510	1.00	96/03/01	Approved	96/03/01	JA511	C:\1995_09\QA\960301-1.MIN
Minutes: Project Meeting	JA511	1.00	96/03/04	Approved	96/03/01		C:\1995_09\QA\960304-1.MIN
Project in Progress Audit	JA512	0.01	96/03/05	Approved	96/03/06		C:\1995_09\QA\9509IP05.PAR
Corrections for JA113	JA513	0.01	96/04/18	Approved	96/12/24	JA525	C:\1995_09\QA\JA513.001
Minutes: Project Meeting	JA514	0.01	96/04/26	Approved	96/05/03	JA516	C:\1995_09\QA\960426-1.MIN
Project in Progress Audit	JA515	0.01	96/04/30	Approved	96/05/18		C:\1995_09\QA\9509IP06.PAR
Minutes: Project Meeting	JA516	0.01	96/05/03	Approved	96/11/07	JA522	C:\1995_09\QA\960503-1.MIN

Document Name	Document Number	Revision Number	Revision Date	Document Status	Date approved by Board	Minute Reference	File Reference
AFRICON 96 Paper	JA120	0.01	96/07/07	Approved	96/12/18	JA524	C:\1995_09\TP\JA120.100
PES 97 Paper	JA121	0.01	96/12/24	Approved	96/12/24	JA525	C:\1995_09\TP\JA121.100
PES 97 Summary	JA122	0.01	96/07/19	Approved	96/12/18	JA524	C:\1995_09\TP\JA122.100
Progress Report 3	JA123	0.01	96/07/25	Approved	96/12/18	JA524	C:\1995_09\TP\JA123.100
Low Level Software Design	JA124	0.01	96/12/24	Approved	96/12/24	JA525	C:\1995_09\TP\JA124.100
Software Test Result Report	JA125	0.01	96/12/24	Approved	96/12/24	JA525	C:\1995_09\TP\JA125.100
Project Front pages	JA126	0.01	96/12/24	Approved	96/12/24	JA525	C:\1995_09\TP\JA126.100
AJPO Integration into Testbed	JA127	0.01	96/12/09	Approved	96/12/18	JA524	C:\1995_09\TP\JA\127.100
AFRICON 96 Presentation	JA128	0.01	96/09/18	Approved	96/12/18	JA524	C:\1995_09\TP\JA128100.PPT
AFRICON 96 Script	JA129	0.01	96/09/19	Approved	96/12/18	JA524	C:\1995_09\TP\JA129.100
Requirements Verification and Validation	JA130	0.01	96/12/24	Approved	96/12/24	JA525	C:\1995_09\TP\JA130.001
ESKOM Documentation	JA131	0.01	96/12/05	Approved	96/12/18	JA524	C:\1995_09\TP\JA131.100
Conclusions and Recommendations for Future Work	JA135	0.01	96/12/24	Approved	96/12/24	JA525	C:\1995_09\TP\JA136.100
Project Bibliography	JA140	0.01	96/12/24	Approved	96/12/24	JA525	C:\1995_09\TP\JA140.100
QUALITY ASSURANCE PRODUCTS							

Document Name	Document Number	Revision Number	Revision Date	Document Status	Date approved by Board	Minute Reference	File Reference
SAUPEC 96 Summary	JA104	1.00	96/03/04	Approved	96/03/04	JA511	C:\1995_09\TP\JA104.100
SAUPEC 96 Paper	JA105	1.00	96/03/04	Approved	96/03/04	JA511	C:\1995_09\TP\JA105.100
PES 96 Summary	JA106	1.00	96/03/04	Approved	96/03/04	JA511	C:\1995_09\TP\JA106.100
PES 96 Paper	JA107	1.00	96/03/04	Approved	96/03/04	JA511	C:\1995_09\TP\JA107.100
AFRICON 96 Summary	JA108	1.00	96/03/04	Approved	96/03/04	JA511	C:\1995_09\TP\JA108.100
SAUPEC 96 Presentation	JA109	1.00	96/03/04	Approved	96/03/04	JA511	C:\1995_09\TP\JA109100.PPT
Product Functional Specification	JA110	1.00	96/11/20	Approved	96/03/04	JA511	C:\1995_09\TP\JA110.100
TESTBED User Reference Manual	JA111	0.01	96/12/09	Approved	96/12/18	JA524	C:\1995_09\TP\JA111.100
Progress Report 2	JA112	0.01	96/03/26	Approved	96/12/18	JA524	C:\1995_09\TP\JA112.100
TESTBED File Format Specification	JA113	0.01	96/12/09	Approved	96/12/18	JA524	C:\1995_09\TP\JA113.100
Interface Specification to DNR	JA114	0.01	96/07/26	Approved	96/12/18	JA524	C:\1995_09\TP\JA114.100
Literature Survey Report	JA115	0.01	96/12/09	Approved	96/12/24	JA525	C:\1995_09\TP\JA115.100
Algorithm Description	JA116	0.02	96/12/24	Approved	96/12/24	JA525	C:\1995_09\TP\JA116.100
High Level Software Design	JA117	0.01	96/12/24	Approved	96/12/24	JA525	C:\1995_09\TP\JA117.100
Software Test Specification	JA118	0.01	96/12/24	Approved	96/12/24	JA525	C:\1995_09\TP\JA118.100
Software Implementation	JA119	0.02	96/12/24	Approved	96/12/24	JA525	C:\1995_09\TP\JA119100.ARJ

2 Master Document List

Document Name	Document Number	Revision Number	Revision Date	Document Status	Date approved by Board	Minute Reference	File Reference
MANAGEMENT PRODUCTS							
Master Document List	JA 001	1.00	96/12/28	Approved	96/03/04	JA511	C:\1995_09\MP\JA001.100
Document Creation Template	JA 002	1.00	96/03/04	Approved	96/03/04	JA511	C:\1995_09\MP\JA002.100
Quality Plan	JA 003	1.00	96/03/04	Approved	96/03/04	JA511	C:\1995_09\MP\JA003.100
Product Description	JA 004	1.00	96/03/04	Approved	96/03/04	JA511	C:\1995_09\MP\JA004.1.00
Project Management Plan	JA 005	1.00	96/09/16	Approved	96/03/04	JA511	C\1995_09\MP\JA005.101
Configuration Management Plan	JA 006	1.00	96/11/20	Approved	96/03/04	JA511	C:\1995_09\MP\JA006.100
Binder Label	JA 008	1.00	96/03/04	Approved	96/03/04	JA511	C:\1995_09\MP\JA008.100
Disk Labels	JA 009	1.00	96/12/18	Approved	96/12/18	JA524	C:\1995_09\MP\JA009.100
TECHNICAL PRODUCTS							
Research Proposal for MSc	JA100	1.00	96/03/04	Approved	96/03/04	JA511	C:\1995_09\TP\JA100.100
Project Description	JA101	1.00	96/03/04	Approved	96/03/04	JA511	C:\1995_09\TP\JA101.100
Progress Report	JA102	1.00	96/03/04	Approved	96/03/04	JA511	C:\1995_09\TP\JA102.100
Mini Project (Test Bed Prototype) Description	JA103	1.00	96/03/04	Approved	96/03/04	JA511	C:\1995_09\TP\JA103.100

- The Product manager, namely Barry Dwolatzky
- All full-time and part-time post-graduate students associated with the SEAL
- Members of the SEAL Management Board
- Head of the Department, Electrical Engineering
- Individuals who will perform internal and external surveillance audits of the SEAL Quality Management System

1.7 Applicable Documents

1.7.1 Standards

a. SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994.

1.8 Assumptions

It is assumed that the reader is familiar with the ISO 9000 series standard for quality systems management.

1.9 Requirements Traceability

This document addresses the following requirements:

a. ISO 9001 (1994) 4.5 Document and Data Control

b. ISO 9001 (1994) 4.16 Quality Records

1 Scope

1.1 Introduction

An essential feature of a quality management system is that it documents the procedures used to implement and maintain the system. This document is the Document Master List which provides a directory of all documents which have the status of Draft, Provisional or Approved.

1.2 Purpose

This Document Master List provides the cross reference to all documents comprising the NODD Project.

1.3 Applicability

This document is an essential reference to all documents supported in the NODD Project.

1.4 Definitions

AFRICON : IEEE Conference in Africa

SAUPEC : South African Universities Power Engineering Conference

IEEE : Institute of Electrical & Electronic Engineering

PES : Power Engineering Society (of IEEE)

NODD : Network Optimisation in Distribution Design

1.5 Notes

The electronic version of a document is assumed to be the master copy and all hard copies are uncontrolled documents, with the exception of signed minutes.

1.6 Audience

The audience for this document comprise the various stakeholders of the SEAL including:

- The product developer, namely Justin Apostolellis

1.00	96/10/31	Updated JA119 in MDL.
1.00	96/11/07	Added JA522. Updated JA521, JA519, JA518 and JA516.
1.00	96/11/13	Added JA125. Updated various dates.
1.00	96/12/09	Updated various dates and added JA523.
1.00	96/12/19	Various documents approved. Added JA524 and JA 009.
1.00	96/12/24	Various documents approved. Added JA525.

Management Authorisation

Version	Date	Status	Management Board Minute Reference
0.01	96/03/04	Approved	JA 511

Change Forecast

This document will be updated each time a new documented element is added into the NODD.

		Added JA501 to list of Quality Assurance Products
0.01	95/08/15	Added JA502 to list of Quality Assurance Products
0.01	95/08/14	Added JA101, JA102 and JA103 to list of Technical Products and JA503 to list of Quality Assurance Products
0.01	95/08/17	Added JA504 and JA505 to the list of QA Products
0.01	95/08/25	Added JA506 to the list of QA Products and JA104 to list of TP products.
0.01	95/10/10	Updated revision date of JA005
0.01	95/10/12	Added JA507 to list of QA products.
0.01	95/12/04	Added JA105 to list of TP products. Data change for JA005.
0.01	95/12/05	Added JA508 to list of QA products.
0.01	95/12/12	Added JA509 to list of QA products.
0.01	96/02/27	Updated JA004. Added JA106, JA107, JA108, JA109
0.01	96/02/28	Added JA110 to list of TP products. Updated JA005
0.01	96/03/01	Added JA510, Updated JA509
1.00	96/03/04	Management and Technical Products promoted to revision 1.00, Added JA511. Approved JA510.
1.00	96/03/05	Added JA512
1.00	96/03/15	Updated revision number of JA006 in MDL
1.00	96/04/18	Updated JA006. Added JA111, JA112, JA113
1.00	96/04/22	Added JA114 to MDL
1.00	96/04/26	Added JA514 to MDL
1.00	96/05/02	Added JA515 to MDL
1.00	96/05/03	Added JA516 to MDL. Updated JA514 in MDL.
1.00	96/07/26	Added numerous entries to Technical Products in MDL .
1.00	96/08/26	Added JA517, updated various dates
1.00	96/08/28	Added JA518.
1.00	96/09/16	Added JA119 and JA127.
1.00	96/09/17	Added JA519
1.00	96/09/19	Added JA128 and JA129
1.00	96/10/15	Updated JA519's approved date.
1.00	96/10/17	Added JA520
1.00	96/10/28	Added JA521

Table of Contents

Table of Contents 2

Change History 3

Configuration Control 3

Document History 3

Revision History 3

Management Authorisation 3

1 Scope 4

1.1 Introduction 4

1.2 Purpose 4

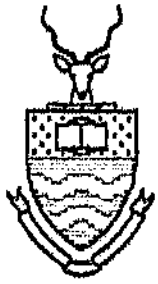
1.3 Definitions 4

1.4 Applicable Documents 4

2 Background Information 9

3 Computer-Based Distribution Design Tools 10

4 Miscellaneous Literature 14



NODD

Literatu.e Survey Report

Technical Product

Version 1.00

Document Status: Approved

6 Configuration Status Accounting

The Project Manager is responsible for ensuring that status of artifacts recorded in the Master Document List corresponds to the:

- Identify of the artifacts in the file directories, and
- The revision numbers are correct.

5 Archiving of Artefacts

5.1 Electronic Artefacts

On project completion an archive copy of all electronic artefacts supporting this projects will be supplied on or more stiffy diskettes using the ARJ.EXE utility. These diskettes will be kept in the official SEAL plastic folders immediately inside the SEAL Project Binder.

5.2 Backup procedure on computers in the SEAL Post-graduate Laboratory Facility

Computers in the SEAL Laboratory Facility are subject to the following back-up procedure:

a. **Daily Backup:** Project files are archived to the stiffy diskettes kept inside the SEAL Project Binders. *(Risk covered: Failure on the hard disk or inadvertant loss or corruption of data.)*

b. **Weekly Backup:** Project files are archived to the SEAL ftp site. This action ensures that these project artifacts are archived to a site outside the SEAL Lab. *(Risk covered: Destruction or loss of the computers in the SEAL Lab by fire, theft, or other disaster.)*

5.3 Backup procedure on computors outside of the Chamber of Mines Building

The archiving of the files on computer system, SEAL server, is subject to weekly tape archival.

The tapes are kept offsite from the location of the resident computer.

5.4 Hardcopy Records

Are maintained in an official SEAL Project Documentation Binder in the Project File in Room CM2221 (SEAL Lab) in the Chamber of Mines Building.

4 Artefact Management

The controls adopted for this project are as follows:

4.1 Active Machine used for supporting the files

SEAL Lab Computer System S-51 is the resident computer.

4.2 Directory Structure

4.2.1 Drive Partition

The C Drive partition is used.

4.2.2 Project Directory

- C:\1995_09\MP - for the storage of the management series documents
- C:\1995_09\TP - used for the storage of technical product
- C:\1995_09\QA - used for the storage of quality assurance products including minutes of review meetings, audits of the system, and correspondence.

AA is a two letter acronym representing the audit type: PI = Project Initiation; IP = Project In-process; PC = Project Closure.

SS is a 2 digit serial number in the range 01 to 99, and starting from 01. SS only applies to audits of type 'IP'.

BBB is the file extension and will be 'PAR', which stands for Project Audit Report.

Example: The third Project In-process Audit Report for a project initiated in 1995, with an ID of 20 would have the following filename: 9520IP03.PAR.

3.1.6 Minutes

The file name will be structured as follows:

YYMMDD-X.BBB

where YYMMDD represents the date of the meeting at which the particular minutes were taken. YY represents the year, MM represents the month and DD the day of the particular meeting.

X is a 1 digit meeting sequence number.

BBB is the file extension and will be 'MIN', which stands for Minutes.

Example: The minutes of the first meeting held on the 4th of March 1995 would be saved in a file with the following filename: 950304-1.MIN.

3.2 Project document sequence numbering

3.2.1 Where the document sequence number comprises 3 digits

- The MP series will be from 001 to 099
- The TP series will be from 100 - 499
- The QA series will be from 500 - 999

Where AAAA is the 4, 3 or 2 letter acronym for this project, which is JA for this project,

XXXX is the 2, 3 or 4 digit sequence number of this document

and ZZZ is the version number of this file, starting from 001 as the initial revision. When a document is baselined as the first version, the most significant digit will represent the version number.

Example: The User Reference Manual (Version 2.00) for the SQM project will have the following file name: SQM003.200.

3.1.3 Code \ line art \ presentation \ image files

The file name will be structured as follows:

AAAXXXYY.BBB

where AAAA is a 2, 3 or 4 letter acronym, not necessarily the project one,

XXX is a 4, 3, or 2 digit serial number

YY is a two digit version number, in the range 01 to 99, and starting from 01.

BBB is the file extension and will be whatever the particular compiler/tool demands.

Example: The third display module (Version 2.4) for the CART project could have the following filename: CDSP0324.PAS, if it was written in Pascal.

Code files are archived together into one file, and will have the same filename structure as above, but the extension will be the named according to the applicable archive type.

3.1.4 Project Audit Reports

The file name will be structured as follows:

YYXXAASS.BBB

where YY is a 2 digit serial number representing the year of project initiation,

XX is a 2 digit serial number representing the project ID according to the SEAL Projects Register (QS 199).

3 Artifact Identification

This section defines a consistent referencing system that shall be used for all artefacts. All artefacts are identified by both an artefact filename followed by a artefact file extension in the following format, as described in the following paragraphs.

<name>.<extension>

3.1 File Naming Convention

3.1.1 Project Products - Multiple word processor types present

The file name will be structured as follows:

AAAXXXYY.ZZZ

Where AAA is 2,3 or 4-letter acronym for this project, which is 'JA' for this particular project,

XXX is the 4, 3, or 2 digit sequence number of this document,

YY is a file type identifier as follows:

- WS - Wordstar
- WP - Word Perfect
- WW - Word for Windows
- AS - pure ascii text
- HC - hard copy file (no source present)
- PS - postscript file

ZZZ is the version number, starting from 001 as the initial version.

Example: QSM001AS.001, and interpreted as the document QSM001 of type AScii and the first version in the sequence.

3.1.2 Project Products - Single word processor format used

The file name will be structured as follows:

AAAAXXXX.ZZZ

2 Documentation Structure

The documents are categorised to reflect the nature of their content. The key categories of documents include:

- management products,
- technical products, and
- quality assurance products.

c. Members of the SEAL Management Board

d. Head of Department of Electrical Engineering

e. Individuals who perform internal and external audits on projects undertaken within the SEAL Quality Management System.

1.6 Applicable Documents

1.6.1 Standards

a. SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994.

1.7 Requirements Traceability

a. ISO9001 (1994) 4.5 Document Control

b. ISO9001 (1994) 4.8 Product Identification and Traceability

c. ISO9001 (1994) 4.13 Control of Non-conforming Product

d. ISO9001 (1994) 4.16 Quality Records

1 Scope

1.1 Introduction

This document describes the configuration management plan applying to all artefacts supporting this project.

Such artefacts will include hardcopy and electronic representations of documents.

This document reviews the procedure which applies to the management and storage of these artefacts.

1.2 Purpose

Configuration items are individual documents and forms in electronic and hardcopy format. They comprise:

- management products and procedures
- technical products, and
- quality products

1.3 Applicability

To be referenced by all individuals who engage in product development for this project.

1.4 Limitations

This configuration management plan is limited to use on one-person projects, or two person projects where one member is the developer (Justin Apostolellis) and the other member is the Product Manager (Barry Dwolatzky).

1.5 Audience

The following comprise the audience for this document:

a. The developer of this product, namely Justin Apostolellis

b. The manager of this product development, namely Barry Dwolatzky

Change History

Configuration Control

Project:	NODD
Title:	Configuration Management Plan
Doc. Reference:	C:\1995_09\MP\JA006
Created by:	Justin Apostolellis
Creation Date:	19 April, 1995

Document History

Version	Date	Status	Who	Saved as:
0.01	95/04/19	Draft	J Apostolellis	C:\1995_09\MP\JA006.001
0.01	95/06/12	Draft	J Apostolellis	C:\1995_09\MP\JA006.001
0.01	95/08/14	Draft	J Apostolellis	C:\1995_09\MP\JA006.001
1.00	96/03/04	Approved	J Apostolellis	C:\1995_09\MP\JA006.100
1.00	96/11/20	Approved	J Apostolellis	C:\1995_09\MP\JA006.100

Revision History

Version	Date	Changes
0.01	95/04/19	New document created from QST12870.101
0.01	95/06/12	Format of dates were corrected
0.01	95/08/14	Sections 3.1.4 and 3.1.5 were included
1.00	96/03/04	Document approved
1.00	96/11/20	Minor changes prior to insertion in dissertation.

Management Authorisation

Version	Date	Status	Project Minute Reference
1.00	96/03/04	Approved	JA511

Change Forecast

This is the cable type id for the cable type that will be used for the service cable segment connecting this consumer to its junction.

3.4.2 Operations:

JConsumerData(newcabletypeId Int, newloadmultfactor JLoadfactor)

This constructor is used to set the values of *cabletypeId* and *loadmultfactor* at creation. They are copied from *newcabletypeId* and *newloadmultfactor* respectively.

3.5 Class name: JInterface

An instance of this class serves as an interface between a client and the optimiser (an instance of *JOptimiser*). A client should create and supply information about the network to an instance of *JInterface*. All nodes are added with *JInterface::addNode(Jnode &, Int)*. The client sends the message *JInterface::optimise(JVoltage)* to tell the instance of *JInterface* to proceed with the optimisation. The client views the resultant configuration by calling *JInterface::viewNodePos(Int, Int&)*.

3.5.1 Has-A Relationships:

networktree * JNode

This data member is a pointer to an array of dynamically created *JNode* objects. This array is created when the *JInterface(Int, JCalculator &, JCableList &)* constructor is used. Its size is *numnodes+1*. The first entry is redundant. The entries in the array are added with the *addNode(JNode &,Int)* member function. The *JInterface* destructor frees the memory occupied by the entries in this array.

treeroot * JNode

This is a pointer to one of the *JNode* objects in the *networktree* array. It is the *JNode* object that represents the junction at the root of the radial network. It is set using the *addNode(JNode &,Int)* member function, to point to the *JNode* object that is added when the value of the *isroot* parameter is nonzero.

calculator * JCalculator

This data member points to an instance of *JCalculator*. It is usually given a value at construction, using the *JInterface(Int, JCalculator &, JCableList &)* constructor. The *JInterface* destructor does not attempt to renew the memory occupied by the instance that *calculator* points to.

numnodes Int

This member function copies the value of *cfvalue* to *cfcurve[numconsumers]*. *JFuncSuccess* is returned if *numconsumers* is not out range, otherwise *JFuncOutOfRange* is returned.

get_ucf(numconsumers Int, ucfvalue float &) : JFuncResultType

This member function copies *ucfcurve[numconsumers]* to the function parameter *ucfvalue*. If *numconsumers* is not out of range, then *JFuncSuccess* is returned, otherwise *JFuncOutOfRange* is returned.

set_ucf(numconsumers Int, ucfvalue float) : JFuncResultType

This member function copies the value of *ucfvalue* to *ucfcurve[numconsumers]*. *JFuncSuccess* is returned if *numconsumers* is not out range, otherwise *JFuncOutOfRange* is returned.

calcCFcurve( ) : JFuncResultType

This member function calculates all coincidence factor values for the *cfcurve* data member using a particular function.

calcUCFcurve( ) : JFuncResultType

This member function calculates all unbalance correction factor values for the *ucfcurve* data member using a particular function.

calcCurOverload(cabtype JCableType &, cur JCurrent) : JFuncResultType

This member function performs a check to see if the current rating associated with *cabtype* exceeds the value of *cur*. If it does then *FuncSuccess* is returned, otherwise *FuncCurExcess* is returned.

3.4 Class name: JConsumerData

Instances of this class are used to store position independent information about consumers. This includes the load multiplication factor of the consumer and the cable type id of the cable type associated with the service cable segment connecting the consumer to a junction.

3.4.1 Has-A Relationships:

loadmultfactor JLoadfactor

This data member is the load multiplication factor of the consumer that is represented by this object.

cabletypeId Int

3.3.2 Operations:

calcVoltdrop(cur JCurrent, len JLength, cabtype JCableType &, thevoltdrop JVoltage &, numconsumers Int) : JFuncResultType

This member function calculates the volt drop in a cable segment and copies the value to *thevoltdrop*. Parameter *len* is the length of the cable segment. Parameter *cabtype* is the cable type. *FuncSuccess* is returned if successful.

calcCurrent(numconsumers Int, numlmfs JLoadfactor, theCurrent JCurrent &) : JFuncResultType

This member function calculates the total current in a cable segment and copies this value in the *theCurrent* parameter. The number of consumers fed by this segment is *numconsumers*. Parameter *numlmfs* is the number of load multiplication factors. *FuncSuccess* is returned if successful.

calcCabCost(len JLength, cabtype JCableType &, cur JCurrent, voltdrop JVoltage, thecost JCost &) : JFuncResultType

This member function calculates the cost of a cable segment and copies it to the *thecost* parameter. Parameter *len* is the cable segment length. Parameter *cabtype* is the cable segment's cable type. Parameter *cur* is current in the cable segment. Its default value is zero, in which case it is ignored. Parameter *voltdrop* is the volt drop across the cable segment. Its default value is zero, in which case it is ignored.

JCalculator(newbreakercurrent JCurrent, newdesignADMD JKwpower, newnomvolts JVoltage)

This constructor is used to set the values of various data members at creation. Parameter *newbreakercurrent* is copied to *breakercurrent* Parameter *newdesignADMD* is copied to *designADMD*. Parameter *newnomvolts* is copied to *nomvolts*. The member functions *calcCFcurve()* and *calcUCFcurve()* are called and if successfully executed then *objectstate* is set to *StateInitialised*, otherwise it is set to *StateUninitialised*.

get_cf(numconsumers Int, cfvalue float &) : JFuncResultType

This member function copies *cfcurve[numconsumers]* to the function parameter *cfvalue*. If *numconsumers* is not out of range, then *JFuncSuccess* is returned, otherwise *JFuncOutOfRange* is returned.

set_cf(numconsumers Int, cfvalue float) : JFuncResultType

Includes data members that hold values for circuit breaker current, ADMD, nominal voltage level, coincidence factors and unbalance correction factors.

3.3.1 Has-A Relationships:

objectstate JObjectState

This data member provides a means of recording the state of an instance of this class. When an instance is created using the *JCalculator(JCurrent, JKwpower, JVoltage)* constructor, calculations for unbalance correction factors and coincidence factors are made. If there is an error during calculation then *objectstate* is set to *StateUninitialised*, otherwise it is set to *StateInitialised*.

designADMD JKwpower

This is the design ADMD to be used for calculation of Diversity factors. It is manipulated with the *get_designADMD()* and *set_designADMD(JKwpower &)* member functions.

breakercurrent JCurrent

This is the breaker current in service cable segments. It is manipulated with the *get_breakercurrent()* and *set_breakercurrent(JCurrent &)* member functions.

cfcurve float[101]

This data member stores an array of values of coincidence factor corresponding to number of consumers between 1 and 100. Entry 0 is redundant. These values are calculated at construction by default equations. They can be changed using the *set_cf(int,float)* member function. Values can be copied using the *get_cf(int, float &)* member function.

ucfcurve float[101]

This data member is an array of values of unbalance correction factors corresponding to number of consumers between 1 and 100. Entry 0 is redundant. These values are calculated at construction by default equations. They can be changed using the *set_ucf(int,float)* member function. Values can be copied using the *get_ucf(int, float &)* member function.

nomvolts JVoltage

The nominal voltage level. It is manipulated using the *get_nomvolts()* and *set_nomvolts(JVoltage &)* member functions.

The number of phases associated with this cable type. Its value is set using *set_numphases(const int value)*. Its value is copied using *get_numphases()*.

cablename char[41]

This is the name of the cable type, stored as a string. It can be up to 40 characters long. It is set using the *set_cablename(const char *)* member function. To receive a pointer to the character array, the *get_cablename()* member function is used.

currentrating JCurrent

The maximum current per phase that this cable type is rated to handle. It is specified in Amps. Its value is set using *set_currentrating(const JCurrent value)*. Its value is copied using *get_currentrating()*.

cabletypeId int

This is a unique identity number, which is associated to this cable type. It is set using *set_cabletypeId(const int value)*. Its value can be copied using *get_cabletypeId()*.

3.2.2 Operations:

JCableType(newname const char *, newcost JCostperlen, newnumphases int, newres JResperlen, newId int, newcur JCurrent)

Constructor that requires specification of the *cablename*, *costperlen*, *numphases*, *resperlen*, *cabletypeId* and *currentrating* attributes associated with this cable type. All of these parameters are copied to their respective data members, with the exception of *newname*, in which case the char array to which this parameter points is copied to *cablename*.

get_cablename( ) : const char *'

Returns a constant char pointer to the *cablename* data member in an instance of this class.

set_cablename(newname const char *) : JFuncResultType

The char array pointed to by *newname* is copied into the *cablename* data member.

3.3 Class name: JCalculator

This class is used to calculate various quantities such as values for volt drops, currents and costs pertaining to cable segments. It encapsulates information that is required to make these calculations, and therefore

to receive such dynamically created memory, otherwise it is set to *StateUnitialised*.

get_cabletype(thecable type JCableType &, theid int) : JFuncResultType

This member function copies *types[theid]* to *thecabletype*. If *theid* is out of range, it returns *FuncOutOfRange*, otherwise it returns *FuncSuccess*.

set_cabletype(newcabletype const JCableType &) : JFuncResultType

This member function copies *newcabletype* to *newcabletype::cabletypeId*. If *newcabletype::cabletypeId* is out of range, it returns *FuncOutOfRange*. If *typesset[newcabletype::cabletypeId]* is equal to 0, then *totalset* is incremented and *typesset[newcabletype::cabletypeId]* is set to 1. It sets *objectstate* to *StateInitialised* if *totalset* is equal to *numtypes*.

get_typesset(theint int &, theid int) : JFuncResultType

This member function copies *typesset[theid]* to *theint*. If *theid* is out of range, it returns *FuncOutOfRange*, otherwise it returns *FuncSuccess*.

3.2 Class name: JCableType

Each cable segment has an associated cable type. Each cable type has an associated specification, in the form of a set of values for the cable type's attributes. This class encapsulates the attributes of a cable type and therefore the specifications of each cable type can be stored in a corresponding instance of this class. Its data members make the storage possible for entities such as name, cost per unit length, resistance per unit length, maximum current rating and number of phases. The data member *JCableType::cabletypeId* represents the identity of the cable type.

3.2.1 Has-A Relationships:

resperlen JResperlen

The resistance per unit length of this cable type specified in ohms per kilometer. Its value is set using *set_resperlen(const JResperlen value)*. Its value is copied using *get_resperlen()*.

costperlen JCostperlen

The cost per unit length of this cable type specified in cost units per meter. Its value is set using *set_costperlen(const JCostperlen value)*. Its value is be copied using *get_costperlen()*.

numphases int

Distribution Design" IEEE Transactions on Power Delivery, Vol. 10, No. 1, February 1995.

[40]Broadwater R, Thompson J, Ellis M, Ng H, Singh N, Loyd D "Application Programmer Interface for the EPRI Distribution Engineering Workstation" IEEE Transactions on Power Systems, Vol. 10, No. 1, February 1995.

[41]Meyer A, Dwolatzky B "The effective computer generation of designs for township electrical distribution", Proceedings of the IEEE Power Engineering Society, Transmission and Distribution Society, Dallas, Texas, September, 1991.

[42]Trudeau D, Hoffman R, Seymour M A "Integrating AM/FM Maps with Distribution SCADA" IEEE Transactions on Power Delivery, Vol. 5, No. 2, April 1990.

[43]Carr J, McCall L V "Divergent Evolution and Resulting Characteristics among the World's Distribution Systems" IEEE Transactions on Power Delivery, Vol. 7, No. 3, July 1992.

[44]Yu D C, Flinn, D G, Krieger R A "Facilitating Engineering Analysis Via a Graphical Database" IEEE Transactions on Power Systems Vol. 10, No. 1, February 1995.

[45]Badenhorst C, Dingley C E, Ferguson I "An Investigation into the use of Intermediate Voltage for Rural Electrification" South African Universities Power Engineering Conference, Pretoria, January 1995.

[46]Wyatt G, Dingley C "A Financial Comparison of Three-Phase versus Single-Phase Rural Electricity Distribution" South African Universities Power Engineering Conference, Pretoria, January 1995.

[26]Goswami S K, Basu S K "A new algorithm for the reconfiguration of distribution feeders for loss minimisation" IEEE Transactions on Power Delivery, Vol. 7, No. 3, July 1992.

[27]Carson M J, Cornfield G 'Design of low-voltage distribution networks' Proceedings of the IEE, Vol. 120, No. 5, May 1973.

[28]Hindi K S, Brameller M I, Gas E 'Design of low-voltage distribution networks: a mathematical programming method' Proceedings of the IEE, Vol. 124, No. 1, January 1977.

[29]Chia C S 'Optimised Distribution Design' Ph. D. thesis, UMIST, 1971.

[30]Snelson B A, Carson M J "Logical design of branched LV distributors" Proceedings of the IEE, Vol. 117, No. 2, February 1970.

[31]Walkden F W "Design of low-voltage distributors" Proceedings of the IEE, Vol. 129, Pt. C, No. 3, May 1982.

[32]Walkden F W "Optimising the design of low-voltage cable networks" Proceedings of the IEE, Vol. 133, No. 1, January 1986.

[33]Nicolson, J M "Object oriented design of cable selection software for low voltage networks", Master's Thesis, University of the Witwatersrand, Johannesburg, April 1993.

[34]Whitaker T, Design Engineer at Johannesburg Electricity Department "Personal Communications", 1996.

[35]Shao J, Rao N D, Zhang Y "An Expert System for Secondary Distribution Design" IEEE Transactions on Power Delivery, Vol. 6, No. 4, October 1991.

[36]Rao N D, Zhang Y "An Intelligent Front End for Secondary Power Distribution System Design" IEEE Transactions on Power Delivery, Vol. 7, No. 2, April 1992.

[37]Sumic Z, Venkata S S, Pistorese T, "Automated underground residential distribution design, Part 1 : Conceptual design", IEEE Transactions on Power Delivery, Vol. 8, No. 2, April 1993.

[38]Sumic Z, Venkata S S, Pistorese T, "Automated underground residential distribution design, Part 2 : Prototype implementation and results", IEEE Transactions on Power Delivery, Vol. 8, No. 2, April 1993.

[39]Yeh E C, Sumic Z, Venkata S S, "APR: A Geographic Information System Based Primary Router for Underground Residential

[14]Gonen T, Ignacio J R R 'Optimal Multi-Stage Planning of Power Distribution Systems' IEEE Transactions on Power Delivery, Vol. PWRD-2, No. 2, April 1987.

[15]Ponnavaiko M, Prakasa Rao K S, Venkata S S 'Distribution System Planning through a Quadratic Mixed Integer Programming Approach' IEEE Transactions on Power Delivery, Vol. PWRD-2, No. 4, October 1987.

[16]Youssef H K, Hackam R 'Dynamic Solution of Distribution Planning in Intermediate Time Range' IEEE Transactions on Power Delivery, Vol. 3, No. 1, January 1988.

[17]Glamocanin V 'Optimal Loss Reduction of Distribution Networks' IEEE Transactions on Power Systems, Vol. 5, No. 3, August 1990.

[18]Ben-Dov E, Harley R G, Seymore W J 'Design of an optimal reticulation system for a residential area' IEEE Transactions on Power Systems, Vol. PWRS-2, No. 1, February 1987.

[19]Willis H L, Powell R W, Vismor T D 'A method of automatically assessing load transfer costs in substation optimisation studies' IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, No.10 October 1985.

[20]Chen J L., Hsu Y Y "An Expert System for Load Allocation in Distribution Expansion Planning" IEEE Transactions on Power Delivery, Vol. 4, No. 3, July 1989.

[21]Hasselfield C W, Wilson P, Penner L, Lau M, Gole A M "An Automated Method for Least Cost Distribution Planning" IEEE Transactions on Power Delivery, Vol. 5, No. 2, April 1990.

[22]Jones A I, Smith B E, Ward D J "Considerations for Higher Voltage Distribution" IEEE Transactions on Power Delivery, Vol. 7. No. 2, April 1992.

[23]Morelato A L, Monticelli A 'Heuristic Search Approach to Distribution System Restoration' IEEE Transactions on Power Delivery, Vol. 4, No. 4, October 1989.

[24]Shirmohammadi D, Wayne H 'Reconfiguration of electric distribution networks for resistive line losses reduction' IEEE Transactions on Power Delivery, Vol. 4, No. 2, April 1989.

[25]Liu C C, Lee J L, Vu K 'Loss Minimisation of Distribution Feeders: Optimality and Algorithms' IEEE Transactions on Power Delivery, Vol. 4, No. 2, April 1989.

[2] Knight U G W 'The Logical Design of Electrical Networks Using Linear Programming Methods' , Proceedings of the IEE, 1960, 107A, pp 306-314.

[3] Boardman J T, Hogg B W 'Computer method for design of electricity-supply networks' Proceedings of the IEE, Vol. 119, No. 7, July 1972.

[4] Crawford D M, Holt S B 'A mathematical optimisation technique for locating and sizing distribution substations, and deriving their optimal service areas' IEEE Transactions on Power Apparatus and Systems, Vol. PAS-94, No. 2 March/April 1975.

[5] Wall D L, Thompson G L, Northcote-Green J E D 'An Optimisation Model for Planning Radial Distribution Networks' ,IEEE Transactions on Power Apparatus and Systems, Vol. PAS-98, No. 3 May/June 1979.

[6] Wall D L, Thompson G L 'A Branch and Bound Model for choosing Optimal Substation Locations', IEEE Transactions on Power Apparatus and Systems, Vol. PAS-100, No. 5 May 1981.

[7] Gonen T, Foote B L 'Distribution-system planning using mixed-integer programming' IEE Proceedings, Vol. 128, Pt. C, No. 2, March 1981.

[8] Sun D I, Farris, D R, Cote P J, Shoults R R, Chen M S 'Optimal Distribution Substation and Primary Feeder Planning via the Fixed Charge Network Formulation', IEEE Transactions on Power Apparatus and Systems, Vol. PAS-101, No. 3 March 1982.

[9] Fawzi T H, Ali K F, El-Sobki S M 'Routing Optimisation of Primary Rural Distribution Feeders', IEEE Transactions on Power Apparatus and Systems, Vol. PAS-101, No. 5 May 1982.

[10]Fawzi T H, Ali K F, El-Sobki S M 'A New Planning Model for Distribution Systems', IEEE Transactions on Power Apparatus and Systems, Vol. PAS-102, No. 9 September 1982.

[11]El-Kady M A 'Computer-Aided Planning of Distribution Substation and Primary Feeders', IEEE Transactions on Power Apparatus and Systems, Vol. PAS-103, No. 6 June 1984.

[12]Boardman J T, Meckiff C C 'A Branch and Bound Formulation to an Electricity Planning Problem' IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, No. 8 August 1985.

[13]Youssef H K, Abu-El-Magd M A 'Novel Optimisation Model for Loong Range Distribution Planning' IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, No. 11 November 1985.

1 Scope

1.1 Introduction

The project entails the application of computer-based automation and optimisation tools to the design of electrical distribution networks. Therefore previously published work on these and related subjects were reviewed through an extensive investigation. The findings of such a literature survey are discussed in this document.

1.2 Purpose

This document presents the findings of the literature survey and serves as a reference to all related documents that were encountered.

1.3 Definitions

AJPO : Automated Junction Position Optimisation

AM/FM : Automatic Mapping / Facilities Management

GIS : Geographic Information System

HV : High Voltage

IV : Intermediate Voltage

LV : Low Voltage

MV : Medium Voltage

SCADA : Supervisory Control And Distribution Automation

1.4 Applicable Documents

1.4.1 Standards

a. SEAL QMS Document Creation Template, QS 002, Revision 0.02, 3 October 1994

b. SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

1.4.2 References

[1] Grimsdale R L, Sinclare P H 'The design of housing-estate distribution systems using a digital computer', Proceedings of the IEE, 1960, 107A, pp 295-305.

Change History

Configuration Control

Project:	NODD
Title:	Literature Survey Report
Doc. Reference:	C:\1995_09\MP\JA115
Created by:	Justin Apostolellis
Creation Date:	22 April 1996

Document History

Version	Date	Status	Who	Saved as:
0.01	96/04/22	Draft	Justin Apostolellis	C:\1995_09\TP\JA115.001
0.01	96/07/18	Draft	Justin Apostolellis	C:\1995_09\TP\JA115.001
0.01	96/08/27	Draft	Justin Apostolellis	C:\1995_09\TP\JA115.001
0.01	96/11/08	Draft	Justin Apostolellis	C:\1995_09\TP\JA115.001
0.01	96/11/09	Draft	Justin Apostolellis	C:\1995_09\TP\JA115.001
0.01	96/12/09	Draft	Justin Apostolellis	C:\1995_09\TP\JA115.001
0.01	96/12/20	Draft	Justin Apostolellis	C:\1995_09\TP\JA115.001
0.01	96/12/24	Approved	Justin Apostolellis	C:\1995_09\TP\JA115.001

Revision History

Version	Date	Changes
0.01	96/04/22	Document was created from template JA002.
0.01	96/07/18	Document format was updated.
0.01	96/08/27	Rearrangement of various sections.
0.01	96/11/08	Various modifications were made.
0.01	96/11/09	Various modifications were made. References arranged.
0.01	96/12/09	Various modifications were made in response to review.
0.01	96/12/20	Various modifications were made in response to next review.
0.01	96/12/24	Document promoted from draft to approved.

Management Authorisation

Version	Date	Status	Management Board Minute Reference
0.01	96/12/24	Approved	JA 525

After an optimisation has been initiated by calling *JInterface::optimise(JVoltage, int, JCost &, JVoltage &)*, this number is set to the number of the element in the *positions* array that corresponds to the position that the optimiser has found for this *JNode*.

objectstate JObjectState

This data member provides a means of recording the state of an instance of this class. When an instance is created using the *JNode(int, int, int, int, int)* constructor, various requests for dynamic memory allocation are made. If there is failure to receive dynamic memory, then the value of *objectstate* is set to *StateMemError*. Otherwise it is set to *StateUninitialised*. The *JPositionData*, *JConsumerData* and sub-tree ids that this object stores are usually subsequently inserted into this instance using respective member functions. Once they have all been successfully inserted, then *objectstate* is set to *StateInitialised*.

totalset int

This data member stores the sum of the numbers of all positions, consumers and sub-trees that have been set by respective member functions.

elementsset * int

This is a pointer to an array of dynamically created *ints*. Its size is *(1 + numpositions + numconsumers + numsubtrees)*. Element 0 is redundant. Elements *(1)* to *(numpositions)* represent positions. Elements *(1+numpositions)* to *(numconsumers+numpositions)* represent consumers. Elements *(1+numpositions+numconsumers)* to *(numpositions+numconsumers+numsubtrees)* represent sub-trees. All elements are set to 0 at construction. Each element is set to 1 if its corresponding entry is set. When all entries are 1, *totalset* should equal *(numconsumers+numpositions+numsubtrees)* and *objectstate* is set to *StateInitialised*.

curIDCS JCurrent

This is the value of the current in the IDCS that feeds the junction represented by this node. It is calculated by *JInterface::calcCurrents(JNode &, int &, JLoadfactor &)*, which recursively calculates currents for all IDCSs.

numconsIDCS int

This is the total number of consumers in the network for which this node is the root. Its value is calculated when *JInterface::calcCurrents()* is called.

3.8.1 Has-A Relationships:

consumers * JConsumerData

This is a pointer to an array of dynamically created *JConsumerData* objects. The number of entries in this array will be the value of *numConsumers +1*. The dynamically created array of objects pointed to by *consumers* is the responsibility of the *JNode* object when destruction occurs.

positions * JPositionData

This is a pointer to an array of dynamically created *JPositionData* objects. The number of entries in this array will be the value is *numPositions + 1*. The *set_positiondata(int, JPositionData &)* and *get_positiondata(int, JPositionData &)* member functions are used to manipulate this data member.

id int

This is the identity number of the *JNode* object.

numconsumers int

This is the number of consumers that are directly connected to this junction. It can be zero or more. It is usually set via the constructor.

numpositions int

This is the total number of possible positions that the junction represented by this node can have. Its value is set at construction.

numsubtrees int

This is the number of sub-trees directly connected to this node. It can be zero or more. Its value is set at construction.

subtreeids int[JmaxnumODCS + 1]

This is an array of *int*s that store the id's of the sub-trees that are connected to this junction. The size of this array is *JmaxnumODCS+1*.

cabletypeid int

This data member holds the value of the cable type id of the cable type corresponding to the IDCS that connects this junction to its parent.

setposition int

This data member stores the highest volt drop in all SCS cable segments in this junction group when it is in the position that is represented by this instance of *JJunctionPosition*.

outputDCSvoltdrop Jvoltage[JmaxnumODCS + 1]

This data member is an array of *JVoltage* elements. Each element represents the voltage drops in one of the ODCSs of the junction group when it is in the position that is represented by this instance of *JJunctionPosition*. The size of the array is (JmaxnumODCS+1). The first entry is redundant. The total number of elements that are actually used is equal to *JJunctionGroup::numsubtrees* of the *JJunctionGroup* that own this instance of *JJunctionPosition*.

3.7.2 Operations:

set_outputDCSvoltdrop(theId int, voltdrop JVoltage &) : JFuncResultType

This member function copies the value of *voltdrop* to *outputDCSvoltdrop[theId]*. If *theId* is out of range, then *FuncOutOfRange* is returned, otherwise it returns *FuncSuccess*.

get_outputDCSvoltdrop(theId int, voltdrop JVoltage &) : JFuncResultType

This member function copies the value of *outputDCSvoltdrop[theId]* to *voltdrop*. If *theId* is out of range, then *FuncOutOfRange* is returned, otherwise it returns *FuncSuccess*.

3.8 Class name: JNode

Objects of this class represent junction groups in an electrical network. They differ from instances of *JJunctionGroup* in that they store primitive information about junction groups, whereas instances of *JJunctionGroup* store information that is directly required by the search algorithm. The data member *JNode::id* denotes the identity of that particular junction. Information about each consumer that is directly connected to the junction is stored by the *JNode::consumers* data member, which is a pointer to *JConsumerData*. Information about each position in which the junction can be placed is stored by the *JNode::positions* data member, which is a pointer to *JPositionData*. The values of the *JNode::id* data members of the junction groups that are directly connected to this junction via its ODCSs are stored by the *subtreeids* data member, which is an array of type *int*. This is how information about the radial structure of the network is stored.

placeHighestCost( ) : JFuncResultType

This member function sets *workselpos* such that it corresponds to minimum volt drop position.

countGroups(numbergroups Int &) : JFuncResultType

This member function finds out from all its sub-trees how many groups they have, by calling this same function. It sums these numbers together and adds itself to this total number. This total number is copied to *numgroups* and *numbergroups*.

setSolution( ) : JFuncResultType

This member function sets *selposition* such that it corresponds to *workselpos*. It calls *setSolution()* for all sub-trees.

placeConfig(configdata int *) : JFuncResultType

This member function uses the information it receives through *configdata* to set *workselpos*. It also calls this same member function in all its sub-trees using the information in *configdata*.

3.7 Class name: JJunctionPosition

An instance of this class stores information about a junction group when its junction is in one of its possible positions. It therefore has data members that store values for the cost of the junction group, IDCS volt drop, ODCS volt drops and largest SCS volt drop. This information pertains to the implicit enumeration criteria that have been formulated for the search algorithm that is used to find the optimal configuration of junction locations.

3.7.1 Has-A Relationships:

cost JCost

This data member stores the total cost of this junction group when it is in the position that is represented by this instance of *JJunctionPosition*.

InputDCSvoltdrop JVoltage

This data member stores the volt drop across the IDCS cable segment of this junction group when it is in the position that is represented by this instance of *JJunctionPosition*.

maxSCSvoltdrop JVoltage

highestcost. From this information it can calculate the maximum volt drops corresponding to each of its positions and therefore ascertain which positions are redundant. It therefore initialises the *numworkpos* and *workpositions* data members for the junction. The values of *workleastvoltdrop* , *workhighestvoltdrop*, *workhighestcost* and *workleastcost* are calculated. If the *JJunctionGroup* has no sub-trees (i.e it is a Terminating Junction), then it does not have to consider its sub-trees. The value of *objectstate* is set to *StateInitialised* when complete. If there is memory allocation failure then it sets it to *StateMemError* and returns *FuncMemError*.

placeAtLeastCost(resultcost JCost &, resultvoltdrop JVoltage &) : JFuncResultType

This member function sets this junction and all sub-tree junctions in their cheapest positions and calculates the maximum volt drop between the root and any consumer. It also sets *resultcost* to the network cost, and *resultdrop* accordingly, when under this configuration. The value of *workselpos* is set appropriately.

placeLeastCost( ) : JFuncResultType

This member function sets *workselpos* such that it corresponds to cheapest position.

placeLessCost( ) : JFuncResultType

This member function sets *workselpos* into the position that corresponds to the next lower junction group cost. It returns *FuncAtLeast* if it is moved into the position of least cost. It returns *FuncNoChange* if it could not change it. Otherwise *FuncSuccess* is returned.

placeHigherCost( ) : JFuncResultType

This member function sets *workselpos* into the position that corresponds to the next highest junction group cost. It returns *FuncAtHighest* if it is moved into the position of highest cost. It returns *FuncNoChange* if it could not change it. Otherwise *FuncSuccess* is returned.

placeAtHighestCost(resultcost JCost &, resultvoltdrop JVoltage &) : JFuncResultType

This member function sets this junction and all the junctions in its sub-trees in their minimum volt drop positions and calculates the maximum volt drop between the root and any consumer. It also sets *resultcost* to the network cost, and *resultdrop* accordingly, when under this configuration. The value of *workselpos* is set appropriately.

set_position(theId Int, theposition const JJunctionPosition &) : JFuncResultType

This member function copies the value of *positions* to *positions[theId]*. If *theId* is out of range then it returns *FuncOutOfRange*, otherwise *FuncSuccess* is returned.

set_subtree(theId Int, thesubtree const JJunctionGroup *) : JFuncResultType

This member function copies *thesubtree* to *subtrees[theId]*. *FuncOutOfRange* is returned if *theId* is out of range, otherwise *FuncSuccess* is returned.

set_workpos(theId Int, theInt const Int &) : JFuncResultType

This member function copies *workpositions[theId]* to *theInt*. If *theId* is out of range then *FuncOutOfRange* is returned, otherwise *FuncSuccess* is returned.

get_workpos(theId Int, theInt Int &) : JFuncResultType

This member function copies *workpositions[theId]* to *theInt*. If *theId* is out of range then *FuncOutOfRange* is returned, otherwise *FuncSuccess* is returned.

optimise(maxvoltdrop JVoltage, resultcost JCost &, resultvoltdrop JVoltage &, configdata Int *) : JOptResultType

This member function performs optimisation in entire tree for which this *JJunctionGroup* is the root. The value of *maxvoltdrop* is the maximum allowed volt drop between the root of the network and any of the consumers. The optimisation result is returned. The values of *resultcost* and *resultvoltdrop* are also set according to the result of the optimisation. The value of *workselpos* in each *JJunctionGroup* is set according to the position that the optimisation has found.

processWorkPos(leastcost JCost &, highestvoltdrop JVoltage &, highestcost JCost &, leastvoltdrop JVoltage &) : JFuncResultType

This member function first calls the same function for all its sub-trees. It receives the value of *highestvoltdrop* from each of its sub-trees, which is the highest volt drop between the root of a sub-tree and any of its consumers when at the position of highest volt drop, at which point the corresponding cost is *lowestcost*. It also receives the value of *leastvoltdrop* from each of its sub-trees, which is the highest volt drop between the root of a sub-tree and any of its consumers when at the position of least volt drop, at which point the corresponding cost is

numgroups Int

This data member stores the total number of junction groups that form the network for which the junction represented by this *JJunctionGroup* is the root. The number includes this junction. It is initialised for all junction groups before optimisation by calling the *countGroups()* member function.

workhighestcost JCost

This data member stores the cost of the entire network, for which this junction is the root, when it is configured to have an associated cost that is higher than all other configurations. It is calculated by *processWorkPos()*.

workhighestvoltdrop JVoltage

This data member stores the largest voltage drop between the root of this junction and all consumers when all junctions within the entire network, for which this junction is the root, are set in the positions that result in overall highest maximum volt drop. It is calculated by *processWorkPos()*.

3.6.2 Operations:

JJunctionGroup(newnumconsumers Int, newnumpositions Int, newnumsubtrees Int)

This constructor is used to set the values of various data members at creation. Parameter *newnumconsumers* is copied to *numconsumers*. Parameter *newnumpositions* is copied to *numpositions*. Parameter *newnumsubtrees* is copied to *numsubtrees*. An array of *numpositions+1 JJunctionPosition* objects is dynamically created and *positions* is set to point to it. The value of *objectstate* is set to *StateMemError* if there was failure to receive such dynamically created memory, otherwise it is set to *StateUnitialised*.

get_position(theid Int, theposition JJunctionPosition &) : JFuncResultType

This member function copies *positions[theid]* to *theposition*. If *theid* is out of range then it returns *FuncOutOfRange*, otherwise *FuncSuccess* is returned.

get_subtree(theid Int, thesubtree JJunctionGroup **) : JFuncResultType

This member function copies *subtrees[theid]* to **thesubtree*. *FuncOutOfRange* is returned if *theid* is out of range, otherwise *FuncSuccess* is returned.

selposition int

This data member stores the position in the *positions* array of the best selected position. It is set after pre-checking and optimisation so that it corresponds to the value of *workselpos* by calling the *setSolution()* function.

numworkpos int

This data member stores the total number of working positions that exist for this junction. A working position is one of the possible positions that have not been found to be redundant. They are therefore used in the search. Its value is calculated when *processWorkPos()* is called.

workpositions * int

This data member is a pointer to a dynamically created array of *ints*, having *numworkpos+1* elements. Each *int* value is an index into the *positions* array. The first entry, *workpositions[1]*, corresponds to cheapest position. The last entry, *workpositions[numworkpos]*, is the most expensive position. The elements are arranged in ascending order of cost. This data member is prepared by *processWorkPos()*.

workselpos int

This data member stores the position in the *workpositions* array of the currently selected position. It is used for setting of positions during optimisation. Its is used by *setSolution()* to set the value of *selposition*.

parent * JJunctionGroup

This data member stores a pointer to the *JJunctionGroup* object that represents the parent of this junction according to the radial structure of the network that is under representation. It is manipulated by the *get_parent()* and *set_parent(JJunctionGroup*)* member functions.

workleastcost JCost

This data member stores the cost of the entire network, for which this junction is the root, when it is configured to have an associated cost that is lower than all other configurations. It is calculated by *processWorkPos()*.

workleastvoltdrop JVoltage

This data member stores the largest voltage drop between the root of this junction and all consumers when all junctions within the entire network, for which this junction is the root, are set in the positions that result in overall lowest maximum volt drop. It is calculated by *processWorkPos()*.

This data member is a pointer to an array of dynamically created *JJunctionPosition* objects. The memory that is occupied is freed in the *JJunctionGroup* destructor. The number of elements in this array is *numpositions* + 1. The elements are manipulated by the *set_position(int, JJunctionPosition &)* and *get_position(int,JJunctionPosition&)* member functions.

subtrees (JJunctionGroup *)[JmaxnumODCS+1]

This data member is an array of pointers to the *JJunctionGroup* objects that are the roots of the sub-trees connected to this instance of *JJunctionGroup*. The size of the array is *JmaxnumODCS+1*. The total number of these pointers that are used by a particular object is equal to *numsubtrees*. The first element is never used. The elements in this array are manipulated with the *set_subtree(int, JJunctionGroup*)* and *get_subtree(int, JJunctionGroup**)* member functions.

numconsumers Int

This data member stores the number of consumers that are directly connected to the junction in the junction group represented by this instance of *JJunctionGroup*. Its value is normally set when the *JJunctionGroup(int, int int)* constructor is used.

numsubtrees Int

This data member stores the number of sub-trees that are directly connected to the junction in the junction group represented by this instance of *JJunctionGroup*. Its value is normally set when the *JJunctionGroup(int, int int)* constructor is used.

numpositions Int

This data member stores the number of possible positions for the junction in the junction group represented by this instance of *JJunctionGroup*. Its value is normally set when the *JJunctionGroup(int, int int)* constructor is used.

objectstate JObjectState

This data member provides a means of recording the state of an instance of this class. When an instance is created using the *JJunctionGroup(int, int, int)* constructor, various requests for dynamic memory allocation are made. If there is failure to receive dynamic memory, then the value of *objectstate* is set to *StateMemError*. Otherwise it is set to *StateUninitialised*. It is set to *StateInitialised* after it has been operated on by *processWorkPos()*.

This function dynamically creates an instance of *JOptimiser* with *numnodes JJunctionGroups*. It then supplies information to all these *JJunctionGroups* in the image of *networktree*. If during the process, there is a memory allocation error, then it deletes all that it has created and returns *FuncMemError*. If *networktree* is not sufficiently initialised, it also deletes all that it has created and returns *FuncNotInitialised*. Otherwise it returns *FuncSuccess*, and sets *(*theoptimiser)* equal to the address of the *JOptimiser* object that it has created. If *overloadstop* is equal to 1 then the process will stop if a cable segment's current exceeds its cable type's current rating. If it is equal to zero, it will carry on regardless.

get_nodeset(theId int, theInt Int &) : JFuncResultType

This member function copies the value of *nodeset[theId]* to *theInt*. If *theId* is out of range, it returns *FuncOutOfRange*, otherwise *FuncSuccess* is returned.

calcCurrents(thenode JNode &, totalnumcons Int &, totalImf JLoadfactor &) : JFuncResultType

This member function is used to calculate the IDCS currents for all the nodes in *networktree*, and stores their values in the corresponding *JNode::curIDCS* data members. It operates on *treeroot*, which causes it to operate on all nodes because of it is a recursive member function.

3.6 Class name: JJunctionGroup

Instances of *JJunctionGroup* represent junction groups in a radial electrical distribution network. They hold information about the costs and volt drops pertaining to IDCS, ODCS and SCS cable segments associated with a particular junction group, at each of the possible locations at which its junction can be placed. They also contain information that is used by the search algorithm that finds the optimal configuration of locations for junctions in the entire network. Each instance has an array of pointers to the instances of *JJunctionGroup* that are the roots its sub-trees. The data member that identifies this array is *JJunctionGroup.:subtrees*. In this manner the structure of the radial network is modeled in a way that parallels the same entity in the algorithm description. The major purpose of the *JJunctionGroup* class is to provide a software entity that models its equivalent algorithm entity. In this manner, the description of the search algorithm is accurately translated into its software equivalent. The member function that performs the optimisation is *JJunctionGroup::optimise(maxvoltdrop JVoltage, resultcost JCost &, resultvoltdrop JVoltage &, configdata Int *) : JOptResultType*.

3.6.1 Has-A Relationships:

positions * JJunctionPosition

This constructor is used to set values for the *calculator*, *cablelist* and *numnodes* data members at creation of an instance of *JInterface*. Parameter *newnumnodes* is copied to *numnodes*. Data member *calculator* is set to point to the address of *newcalculator*. Data member *cablelist* is set to point to the address of *newcablelist*. An array of *JNode* objects is created and is pointed to by *networktree*. Its size is *numnodes +1*. An array of *Int*s is created and pointed to by *nodesset*. Its size is also *numnodes +1*. All these entries in this array are set to zero at construction. The value of *totalset* is set to zero. The *objectstate* data member is set to *StateMemError* if failure to receive dynamically allocated memory occurs during construction. Otherwise *objectstate* is set to *StateUninitialised* if *newnumnodes* is greater than 0.

addNode(newnode JNode &, isroot Int) : JFuncResultType

This member function copies *newnode* into *networktree[newnode::id]*. If the value of *isroot* is 1 then it makes *treeroot* point to the *networktree[newnode::id]*. *FuncMemError* is returned if there was a failure to receive memory during execution. *FuncOutOfRange* is returned if the *newnode::id* is less than 1 or greater than *numnodes*. If *newnode::objectstate()* is equal to *StateInitialised* and *nodesset[newnode::id]* is 0 then it increments *totalset* and sets *nodesset[newnode::id]* to 1. If *totalset* is equal to *numnodes* and *cablelist* and calculator are fully initialised then *objectstate* is set to *StateInitialised*.

optimise(maxvoltdrop JVoltage, overloadstop Int , resultcost JCost &, resultvoltdrop JVoltage &) : JOptResultType

This member function performs the optimisation only if *objectstate* is equal to *StateInitialised* and returns an appropriate result code, otherwise it returns *OptNotInitialised*. If *overloadstop* is equal to 1 then the optimisation will not occur if a cable segment's current exceeds its cable type's current rating, if it is equal to zero, it will optimise regardless. The default value is zero. The parameter *maxvoltdrop* holds the value of the maximum volt drop between the root of the network and any of the consumers. The resultant cost is returned through *resultcost*. The resultant maximum volt drop is returned through *resultvoltdrop*.

viewNodePos(nodeId Int, position Int &) : JFuncResultType

This member function allows the position of an individual node to be viewed after an optimisation. The node's id is given as an input via *nodeId*. If *nodeId* is within range, then the position number in which that node has been is copied to *position* and *FuncSuccess* is returned, otherwise *FuncOutOfRange* is returned.

makeOptimiser(theoptimiser JOptimiser **, overloadstop Int) : JFuncResultType

This data member stores the number of *JNode* objects in the network. It is set when the *JInterface(int, JCalculator &, JCableList &)* constructor is used.

cablelist * JCableList

This data member points to an instance of *JCableList*. It is usually given a value at construction, using the *JInterface(int, JCalculator &, JCableList &)* constructor. The *JInterface* destructor does not attempt to renew the memory occupied by the instance that *cablelist* points to.

objectstate JObjectState

This data member provides a means of recording the state of an instance of this class. When an instance is created using the *JInterface(int, JCalculator &, JCableList &)* constructor, various requests for dynamic memory allocation are made. If there is failure to receive dynamic memory, then the value of *objectstate* is set to *StateMemError*. Otherwise it is set to *StateUninitialised*. The *JNode* objects that the instance of *JInterface* stores are usually subsequently inserted using the *addNode(JNode &,int)* member function. Once all *JNode* objects have been successfully inserted and the value of *cablelist::objectstate* and *calculator::objectstate* are equal to *StateInitialised*, then *objectstate* is set to *StateInitialised*.

totalset int

This data member stores the number of initialised nodes that have been added to this instance of *JInterface* with the *addNode(JNode &,int)* member function.

nodesset * int

This is a pointer to an array of dynamically created *int*s. When an instance of *JInterface* is created using the *JInterface(int, JCalculator &, JCableList &)* constructor, then this data member is set to point to an array of dynamically created *int*s. The size of the array is *numnodes+1*. This dynamically created array of *int*s is used to keep track of which *JNodes* in the array pointed to by *networktree* have been set using the *addNode(JNode &,int)* member function. If a *JNode object* is set then its corresponding entry in the *nodesset* array is set to 1, otherwise it remains zero.

3.6.2 Operations:

JInterface(newnumnodes int, newcalculator JCalculator &, newcablelist JCableList &)

4 Software Usefulness Testing Results

This section covers tests 5.1 and 5.2

4.1 TEST 5-1 Underground 1

The cost of cables for the original network is 24482 Cost Units. The original network has an associated maximum consumer volt drop of 17.48 volts and therefore the network was optimised with a slightly larger value for largest allowed maximum consumer volt drop(17.5). The cost of cables for the optimised network is 22786.6 Cost Units. Therefore the optimisation yielded a 6.93 % saving. The optimisation in this test yielded the network shown on the left of the map in figure 4.1.

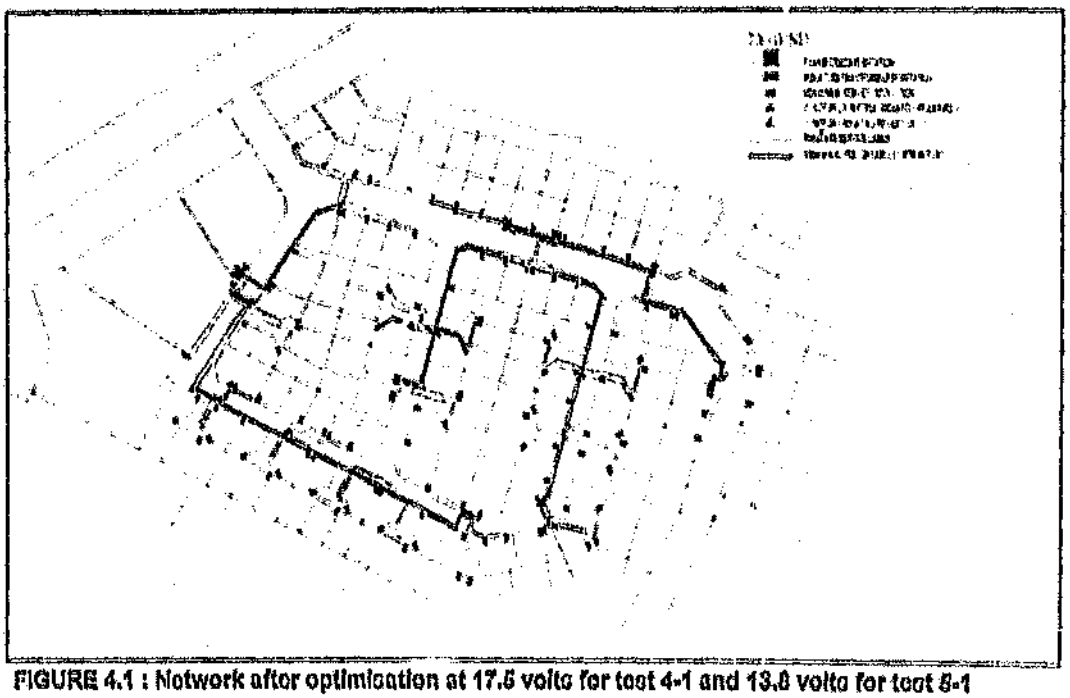


FIGURE 4.1 : Network after optimisation at 17.5 volts for test 4-1 and 13.8 volts for test 5-1

The same network was optimised with the default largest allowed maximum consumer volt drop which is 13.8 volts. The cable costs for the optimised network is 23348 Cost Units. Therefore the optimisation yielded a 4.63 % saving. The optimised network also has an associated maximum consumer volt drop of 16.27 volts which is 1.21 volts less than the original network, but it still causes a violation of the maximum volt drop constraint.

3 Reliability Testing Results

This section covers tests 4.1 to 4.3. The expected and actual results are p[illegible]nted in table 3.1

Test	result type	objectstate
4.1(a)	expected	20
	actual	20
4.1(b)	expected	20
	actual	20
4.1(c)	expected	10
	actual	10
4.1(d)	expected	20
	actual	20
4.2(a)	expected	20
	actual	20
4.2(b)	expected	10
	actual	10
4.3(a)	expected	20
	actual	20
4.3(b)	actual	20
	expected	20
4.3(c)	actual	10
	expected	10
4.3(d)	actual	20
	expected	20

TABLE 3.1 Reliability Testing Results

The identical values for expected and actual entities illustrates that the reliability testing reveals that the code is reliable.

2 Correctness Testing Results

This section covers tests 3.1 to 3.7. The expected and actual results are presented in table 2.1

Test	result type	result value	result volt drop	result cost	junction 1 position	junction 2 position	junction 3 position	junction 4 position	junction 5 position
1(a)	expected	2	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	actual	2	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1(b)	expected	2	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	actual	2	N/A	N/A	N/A	N/A	N/A	N/A	N/A
2	expected	10	14.666	13702.46	2	1	1	2	2
	actual	10	14.666	13702.46	2	1	1	2	2
3	expected	20	13.304	13038.4	2	1	1	2	3
	actual	20	13.304	13038.4	2	1	1	2	3
4	expected	30	13.14266	14022	1	1	2	1	3
	actual	30	13.14266	14022	1	1	2	1	3
6	expected	40	14.666	13702.46	2	1	1	2	2
	actual	40	14.666	13702.46	2	1	1	2	2
6	expected	50	13.304	13038.4	2	1	1	2	3
	actual	50	13.304	13038.4	2	1	1	2	3
7	expected	60	13.14266	14022	1	1	2	1	3
	actual	60	13.14266	14022	1	1	2	1	3

TABLE 2.1 Correctness Testing Results

The identical values for expected and actual entities illustrates that the correctness testing reveals that the code is correct.

1 Scope

1.1 Introduction

The accuracy and functionality of the software has been tested to validate that it addresses the requirements that were established prior to development.

1.2 Purpose

This document specifies the expected and actual results of the test procedures that were conducted to test the software. It also discusses the performance of the software.

1.3 Applicability

The software tool that these tests pertain to is the AJPO tool, which is a product of the NODD project.

1.4 Applicable Documents

1.4.1 Specifications

a. Software Test Specification, JA118, Revision 1.00, 24 December 1996

1.4.2 Standards

a. SEAL QMS Document Creation Template, QS 002, Revision 0.02, 3 October 1994

b. SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994

Change History

Configuration Control

Project:	NODD
Title:	Software Test Result Report
Doc. Reference:	C:\1995_09\YP\JA125
Created by:	Justin Apostolellis
Creation Date:	1 November1996

Document History

Version	Date	Status	Who	Saved as:
0.01	96/11/01	Draft	Justin Apostolellis	C:\1995_09\MP\JA125.001
0.01	96/11/04	Draft	Justin Apostolellis	C:\1995_09\MP\JA125.001
0.01	96/11/19	Draft	Justin Apostolellis	C:\1995_09\MP\JA125.001
0.01	96/11/20	Draft	Justin Apostolellis	C:\1995_09\MP\JA125.001
1.00	96/12/24	Draft	Justin Apostolellis	C:\1995_09\MP\JA125.100

Revision History

Version	Date	Changes
0.01	96/11/01	Document was created from template JA002. Chapter 2 was created.
0.01	96/11/04	Results in table 2.1 in Chapter 2 were updated. Chapter 3 was added.
0.01	96/11/19	Chapter 4 updated.
0.01	96/11/20	Minor changes made to chapter 4.
1.00	96/12/24	Document Status promoted from draft to approved.

Management Authorisation

Version	Date	Status	Management Board Minute Reference
0.01	96/12/24	Approved	JA 625

Change Forecast

Table of Contents

Table of Contents ii

Change History iii

Configuration Control iii

Document History iii

Revision History iii

Management Authorisation iii

Change Forecast iii

1 Scope 1

1.1 Introduction 1

1.2 Purpose 1

1.3 Applicability 1

1.4 Applicable Documents 1

2 Correctness Testing Results 2

3 Reliability Testing Results 3

4 Software Usefulness Testing Results 4

4.1 TEST 6-1 Underground 1 4

4.2 TEST 6-2 Underground 2 5



NODD

Software Test Result Report

Technical Product

Version 1.00

Document Status: Approved

set_ODCSlength(ODCSnumber int, newlength JLength) : JFuncResultType

This member function copies the value *thelength* to *ODCSlength[ODCSnumber]*. *FuncOutOfRange* is returned if *ODCSnumber* is out of range, otherwise *FuncSuccess* is returned.

get_SCSlength(SCSnumber int, thelength JLength &) : JFuncResultType

This member function copies the value of *SCSlength[SCSnumber]* into *thelength*. *FuncOutOfRange* is returned if *SCSnumber* is out of range, otherwise *FuncSuccess* is returned.

get_ODCSlength(ODCSnumber int, thelength JLength &) : JFuncResultType

This member function copies the value of *ODCSlength[ODCSnumber]* into *thelength*. *FuncOutOfRange* is returned if *ODCSnumber* is out of range, otherwise *FuncSuccess* is returned.

■

FuncMemError is returned if copying failed due to memory allocation failure. *FuncSuccess* is otherwise returned.

optimise(maxvoltdrop JVoltage, resultcost JCost &, resultvoltdrop JVoltage &) : JOptResultType

This member function is called to perform the optimisation such that *maxvoltdrop* is the maximum allowed volt drop between root and any of the consumers. The optimisation result is returned. The resultant cost is returned through *resultcost*. The resultant maximum volt drop is returned through *resultvoltdrop*.

3.10 Class name: JPositionData

This class encapsulates information about cable segment lengths for a junction group. Each instance holds information about a single position for a junction in a single junction group. It contains the values of all the IDCS, ODCS and SCS lengths in the junction group.

3.10.1 Has-A Relationships:

IDCSlength JLength

This stores the length of the IDCS connected to this junction when the junction is in the position represented by this instance of *JPositionData*.

ODCSlength JLength[JmaxnumODCS + 1]

This is an array of *JLength* entities. Its size is *JmaxnumODCS+1*. Each entry stores the value of an ODCS length. Only the entries between 1 and *numsubtrees* in the *JNode* object that owns the instance of *JPositionData* are used.

SCSlength JLength[JmaxnumSCS + 1]

This is an array of *JLength* entities. Its size is *JmaxnumSCS+1*. Each entry stores the value of an SCS length. Only the entries between 1 and *numconsumers* in the *JNode* object that owns the instance of *JPositionData* are used.

3.10.2 Operations:

set_SCSlength(SCSnumber Int, newlength JLength) : JFuncResultType

This member function copies the value *thelength* to *SCSlength[SCSnumber]*. *FuncOutOfRange* is returned if *ODCSnumber* is out of range, otherwise *FuncSuccess* is returned.

This is a pointer to an array of dynamically created *JJunctionGroup* objects that make up the network of junctions. The number of elements in this array is *numGroups+1*. The *JOptimiser* destructor frees the memory occupied by these dynamically created objects.

numgroups Int

This is the number of *JJunctionGroup* objects in the network that are contained by the *JOptimiser*. This value is usually set with the *JJunctionGroup(Int)* constructor.

rootjunction * JJunctionGroup

This is a pointer to the *JJunctionGroup* object which is the root of the tree of junction groups. Its value is manipulated with the *set_rootjunction(JJunctionGroup *)* and *get_rootjunction()* member functions.

objectstate JObjectState

This data member provides a means of recording the state of an instance of this class. When an instance is created using the *JOptimiser(Int)* constructor, various requests for dynamic memory allocation are made. If there is failure to receive dynamic memory, then the value of *objectstate* is set to *StateMemError*. Otherwise it is set to *StateInitialised*.

3.9.2 Operations:

JOptimiser(newnumgroups Int)

This constructor dynamically creates an array of *JJunctionGroup* objects that *junctiongroups* is set to point to. The value of *objectstate* is set to *StateMemError* if memory error occurs, otherwise it is set to *StateUninitialised*. It becomes *StateInitialised* after *processWorkPos()* has been called. Therefore it is the client's responsibility to make sure that all *JJunctionGroup* objects are properly initialised before *processWorkPos()* is called.

get_group(theId Int, thegroup JJunctionGroup **) : JFuncResultType

This member function sets **thegroup* equal to *&(junctiongroups[theId])*. *FuncOutOfRange* is returned if *theId* is out of range, otherwise *FuncSuccess* is returned.

set_group(theId Int, thegroup const JJunctionGroup &) : JFuncResultType

This member function copies the value of *thegroup* to *junctiongroups[theId]*. *FuncOutOfRange* is returned if *theId* is out of range.

less than or equal to *JNode::numconsumers* and greater than zero. Otherwise *FuncOutOfRange* is returned.

set_consumerdata(consumernum Int, theconsumer JConsumerData &) : JFuncResultType

This member function copies the value of *theconsumer* to *consumers[consumernum]*. It returns *FuncSuccess* if the value of *consumernum* is less than or equal to *JNode::numconsumers* and greater than zero. Otherwise *FuncOutOfRange* is returned. The values of *JNode::elementsset*, *JNode::totalset* and *JNode::objectstate* are appropriately updated.

get_positiondata(positionnum Int, theposition JPositionData &) : JFuncResultType

This member function copies the value of *positions[positionnum]* to *theposition* and returns *FuncSuccess* if the value of *positionnum* is less than or equal to *JNode::numpositions* and greater than zero. Otherwise *FuncOutOfRange* is returned.

get_subtreeid(subtreenum Int, thesubtreeId Int &) : JFuncResultType

This member function copies the value of *subtreeids[subtreenum]* to *thesubtreeid* and returns *FuncSuccess* if the value of *subtreenum* is less than or equal to *JNode::numsubtrees* and greater than zero. Otherwise *FuncOutOfRange* is returned.

get_elementsset(theInt Int &, theId Int) : JFuncResultType

This member function copies the value of *elementsset[theId]* to *theInt* and returns *FuncSuccess* if the value of *theId* is within range. Otherwise *FuncOutOfRange* is returned.

3.9 Class name: JOptimiser

An instance of *JInterface* creates an instance of *JOptimiser* to perform the junction position optimisation. The *JInterface* object supplies it with information via the *JOptimiser::set_group(Int, JJunctionGroup &)* member function, which copies instances of *JJunctionGroup* to the array identified by *JOptimiser::junctiongroups*. Optimisation proceeds when the *JOptimiser::optimise(JVoltage, JCost&, JVoltage&)* member function is called.

3.9.1 Has-A Relationships:

junctiongroups * JJunctionGroup

ImflDCS JLoadfactor

This is the total load multiplication factor in the netwrk for which this node is the root. Its value is calculated when *JInterface::calcCurrents()* is called.

3.8.2 Operations:

JNode(newId Int, newnumconsumers Int, newnumpositions Int, newnumsubtrees Int, newcabletypeId Int)

This constructor is used to set the values of various data members at creation. Parameter *newId* is copied to *Id*. Parameter *newnumconsumers* is copied to *numconsumers*. Parameter *newnumpositions* is copied to *numpositions*. Parameter *newnumsubtrees* is copied to *numsubtrees*. Parameter *newcabletypeId* is copied to *cabletypeId*. It dynamically creates an array of *(numpositions+1) JPositionData* objects and sets *positions* to point to it. It dynamically creates an array of *(numconsumers+1) Ints* and sets *totalset* to points to it. It dynamically creates an array of *(numpositions+numconsumers+numsubtrees+1) Ints* and sets *elementsset* to points to it. Data member *objectstate* is set to *StateMemError* if there was failure to receive such dynamically created memory, otherwise it is set to *StateUnitialised*.

set_subtreeId(subtreenum Int, thesubtreeId Int) : JFuncResultType

This member function copies the value of *thesubtreeId* to *subtreeIds[subtreenum]*. It returns *FuncSuccess* if the value of *subtreenum* is less than or equal to *JNode::numsubtrees* and greater than zero. Otherwise *FuncOutOfRange* is returned. The values of *JNode::elementsset*, *JNode::totalset* and *JNode::objectstate* are apropriately updated.

set_positiondata(positionnum Int, theposition JPositionData &) : JFuncResultType

This member function copies the value of *theposition* to *positions[positionnum]*. It returns *FuncSuccess* if the value of *positionnum* is less than or equal to *JNode::numpositions* and greater than zero. Otherwise *FuncOutOfRange* is returned. The values of *JNode::elementsset*, *JNode::totalset* and *JNode::objectstate* are appropriately updated.

get_consumerdata(consumernum Int, theconsumer JConsumerData &) : JFuncResultType

This member function copies the value of *consumers[consumernum]* to *theconsumers* and returns *FuncSuccess* if the value of *consumernum* is

2 Procedure

When the functional requirements have been identified in the Product Description, map these into the column 2. These requirements provide the baseline for verification and validation activity across the columns.

	Section reference in:					
Requirement Reference	Product Description (JA004)	Product Functional Specification (JA 110)	Algorithm Description (JA116)	High Level Software Design (JA117)	Software Test Specification (JA 118)	Software Test Result Report (JA 126)
3.1.	Automatically find optimal network configurations by searching for optimal junction locations.	Requirement addressed in Sections 2 and 5.	Requirement addressed in Sections 5.2 and 6.5	Requirement addressed in Sections 4 and 8.2.	Requirement addressed in Section 3.	Requirement addressed in Section 2.
3.2.	Suitable interface to provide the required input information.	Requirement addressed in Section 4.	Requirement addressed in Section 6.4.	Requirement addressed in Sections 4 and 8.1.	Requirement addressed in Section 3.	Requirement addressed in Section 2.
3.3.	Suitable interface to receive the output information.	Requirement addressed in Section 6.	Requirement addressed in Section 6.6	Requirement addressed in Sections 4 and 8.3.	Requirement addressed in Section 3.	Requirement addressed in Section 2.

1.6 Assumptions

It is assumed that the reader is familiar with the ISO 9000 series standard for quality systems management.

1.7 Requirements Traceability

This document addresses the following requirements:

a. ISO 9001 (1994) 4.4 Design Control

1 Scope

1.1 Purpose

This document provides the cross reference to all requirements identified in such documents as: Product Description, Product Functional Specification, Algorithm Description, High Level Software Design, Software Test Specification, Software Test Result Report.

1.2 Applicability

This document is used to support verification and validation activity on software development.

1.3 Definitions

AJPO : Automatic Junction Position Optimisation

1.4 Audience

The audience for this document comprise the various stakeholders of the SEAL, including:

- The product developer, Justin Apostolellis
- The Product manager, Barry Dwolatzky
- All full-time and part-time post-graduate students associated with the SEAL
- Members of the SEAL Management Board
- Head of the Department, Electrical Engineering
- Individuals who will perform internal and external surveillance audits of the SEAL Quality Management System

1.5 Applicable Documents

1.5.1 Standards

a. SEAL QMS Document Layout, Presentation and Typesetting Guide, QS 003, Revision 1.00, 3 October 1994.

Change History

Configuration Control

Project:	NODD
Title:	Requirements Verification and Validation
Doc. Reference:	C:\1995_09\TP\JA130
Created by:	Justin Apostolellis
Creation Date:	20 November 1996

Document History

Version	Date	Status	Who	Saved as:
0.01	96/11/20	Draft	J Apostolellis	C:\1995_09\TP\JA130.001
1.00	96/12/24	Approved	J Apostolellis	C:\1995_09\TP\JA130.100

Revision History

Version	Date	Changes
0.01	96/11/20	New document created using QST324-90.
1.00	96/12/24	Document Status promoted from draft to approved.

Management Authorisation

Version	Date	Status	Management Board Minute Reference
0.01	96/12/24	Approved	JA 525

Change Forecast

This document will be updated each time new requirements are identified or changed in existing documents.

Table of Contents

Table of Contents 2

Change History 3

Configuration Control 3

Document History 3

Revision History 3

Management Authorisation 3

Change Forecast 3

1 Scope 4

1.1 Purpose 4

1.2 Applicability 4

1.3 Definitions 4

1.4 Audience 4

1.5 Applicable Documents 4

1.6 Assumptions 5

1.7 Requirements Traceability 5

2 Procedure 6



NODD

Requirements Verification And Validation

Technical Product

Version 1.00

Document Status: Approved

4.2 TEST 5-2 Underground 2

The cost of cables for the original network is 25449 Cost Units. The network was optimised with a largest allowed maximum consumer volt drop of 13.8 volts. The cost of cables for the optimised network is 24738.5 Cost Units. Therefore the optimisation yielded a 2.79 % saving. The optimised network is also the cheapest possible network and therefore the optimiser did not have to perform a search to find the solution. The optimisation in this test yielded the network shown on the right of the map in figure 4.1.

■

Author: Apostolellis Justin.
Name of thesis: The production of software that aids in the design of low voltage distribution networks by optimising the locations of junctions.

PUBLISHER:
University of the Witwatersrand, Johannesburg
©2015

LEGALNOTICES:

Copyright Notice: All materials on the University of the Witwatersrand, Johannesburg Library website are protected by South African copyright law and may not be distributed, transmitted, displayed or otherwise published in any format, without the prior written permission of the copyright owner.

Disclaimer and Terms of Use: Provided that you maintain all copyright and other notices contained therein, you may download material (one machine readable copy and one print copy per page)for your personal and/or educational non-commercial use only.

The University of the Witwatersrand, Johannesburg, is not responsible for any errors or omissions and excludes any and all liability for any errors in or omissions from the information on the Library website.