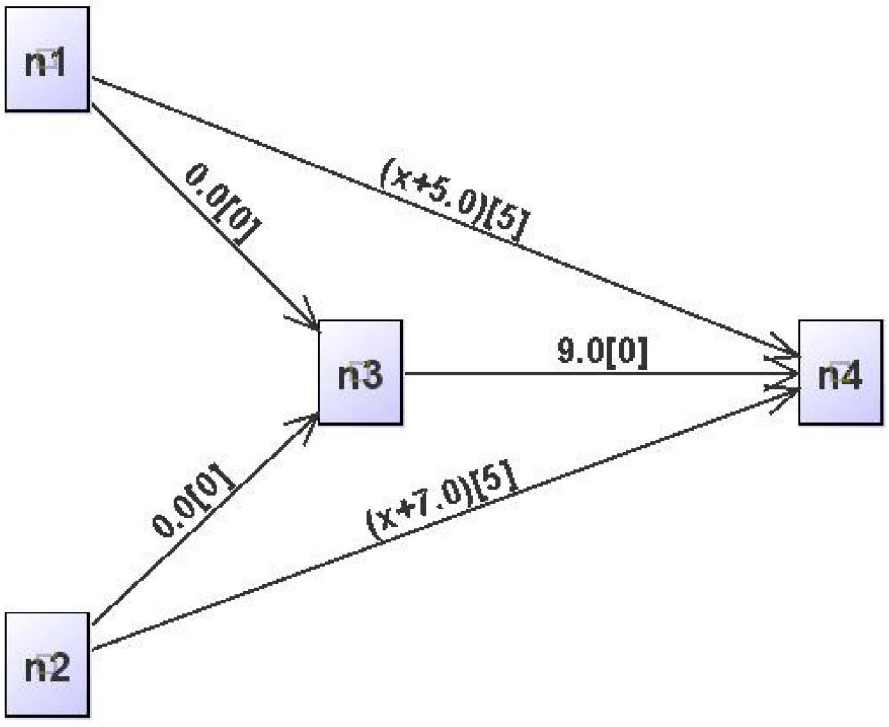
TNA

Traffic Network Analyzer



TU Delft
Delft University of Technology

Document:	**Bachelor Project Process Report**
Course Code and Name:	**IN3405 Bachelorproject**
Date:	**January 2009**
Author:	**Khaled Tokhi 1111647**
Department:	**Software Engineering**
Faculty:	**Technical Informatics**
Institution:	**University of Technology Delft**
Bachelor Commission:	**Dr. Hans Melissen (Information Systems and Algorithms)**
	Ir. Bernard Sodyer (Information Systems Design Department)

TU Delft
Delft University of Technology

Table of Contents

Foreword

This document is written as a partial requirement for the Bachelor Project Degree IN3700, which is conducted out under combined guidance and supervision of Dr. J.B.M. Melissen from the Information Systems and Algorithms department and Ir. B.R. Sodoyer from the Information Systems Design Department at Delft University of Technology. I want to thank both the coordinators for their precious guidance and advice that I received from them through the project. Specially their understanding of my personal circumstances which lead to a long pause in the project.

September 2008 Delft
Khaled Tokhi

Executive Summary

TNA stands for Traffic Network Analyzer which is a software program that is able to calculate the optimal route in a multi-commodity (traffic) road network using Minimum Delay Flow deviation algorithm. TNA can be used as a tool to help understanding where traffic jams occur in road networks it can also be useful in the field of optimization and logistics. The main user group for TNA software was identified as traffic authorities who are responsible for organization of the road networks. TNA uses as input topology of the network, delay function of network links and total required inflow. Based on the given input TNA calculates the minimum delay using Minimum Flow Deviation algorithm and provides the result set by specifying which route will have the shortest delay. Using TNA software it's users can get an overview of the performance of a network, given a certain amount of total inflow and also get the optimal route to flow commodities from a given sauce to destination. Having these results traffic network authorities or anyone using TNA will get overview of the performance of the network.

TNA software was build using a classic waterfall in combination with extremeProgramming methodology. This report gives a description of the activities which took place during the course of project. Each chapter of the report mainly represents a major phases of the project. Section 2 describes the project analysis phase from the point of view of the coordinators. Section 3 describes the process of designing and requirements gathering and specifications. Section 4 the implementation related aspects of the software will be explained followed by testing activates in section 5. The report ends by providing suggestion on further development and extension of TNA software. All the project related documents are attached as appendixes at the end of report.

1.Introduction

This section gives a brief introduction to the context in which this project has been carried out by briefly explaining the actual background and motivation this is followed by details on the project domain and organization.

1A.Tragedy of Commons (Background)

Nowadays one of the major problems in traffic networks is the selfish behavior of the commuters who use the traffic road networks. Basically every individual commuter wants to reach his or her individual destination in the shortest possible time following the shortest route without regard to the effects that his or her individual behavior has on other users of the road network. This kind of behavior generally leads to a situation in which some roads are overcrowded because the number of commuters exceeds the road capacity and thus causing traffic jams where every commuter is forced to spend more time than in the roads which are not overloaded.

Thus, in order to balance the effects of the selfish behavior of commuters, traffic authorities need to take some measures which will force commuters to use alternative routes so that an optimum travel time is achieved by every individual user of the road network as a whole. One such measure can be to impose some sort of taxes on those routes which are mostly congested because apparently they appear to have a large number of commuters as they provide the shortest route from the commuter's source to destination. The taxing schemes are not aimed at reducing the number of commuters or road users but at deviating the commuters' route and thus at preventing them from taking a particular route to reach their destination.

But before imposing such taxes traffic authorities need to identify the links (i.e. the roads) which are mostly over congested and to establish what is the optimum way to tax these over congested roads. For this they need to compute the user equilibrium and system optimum of a traffic network which seems to be over congested.

The software tool developed for this project is called Traffic Network Analyzers or TNA. TNA allows a user friendly input like network topology, specifications of nodes, properties of edges (links) such as the total inflow per node, delay function of edges, different types of commodities the total required flow per commodity etc. Using the user given input TNA would be able to compute an optimal flow pattern for a static network. Software design, implementation and testing is done using standard software engineering principles and techniques.

1B.Project Domain

The domain in which this project is carried out is a combination of optimization methods and software engineering principles and techniques, the objective being to compute, analyze and solve static network flow problems.

Therefore the main project objectives can be categorized into two groups. Firstly it is necessary to study of optimization algorithms for computing a optimal flow pattern for network with static flows. Secondly the implementation of an algorithm in n computers program using Java as programming language and object oriented software engineering techniques and principles.

1C.Project planning

This section describes the planning and organization of the project. During the project I tried to stick to the guidelines TU-Delft laid down for bachelor projects as much as possible. The officially designated time limit for this project was 12 weeks, but because of interruptions due to exams and holiday periods the tasks could not be achieved in a continuous manner. This project started officially in second week of July 2007 and continued until 23rd of July 2007 and a break was taken due to holidays and the exams afterwards, in September the project was supposed to continue until end of October but due to an accident it was halted for one academic year till July 2008. The table below shows the initial time schedule of the project which was agreed and reviewed by me, the project coordinators and supervisors.

Table1:

Week	Project Week	Project Phase	Deliverables (Date)
45 (06.11.2006)	5	Software Design	1.SDD 1.1 (7.11.2006) 2.Core Algorithm for calculating U.O and S.E (8.11.2006) 3.RAD version 1.2(9.11.2006) 4.Mid Term Evaluation Report (10.11.2006)
		November	No Activity
		December	No Activity
		January	No Activity

6 (05.02.2007)	6	Implementation	1.Plan of Approach version 1.5 2.Table of Content of Project Report Test Plan(5.02.2007)
7	7	Implementation	
8	8	Debugging/Testing of code	Beta Version of TNA
9	9	Testing of Program	RC1 of TNA
10	10	Project Report	Final Version
11	11	Prepare A Draft Report of Project	First version of Project Report
12	12	Prepare Project Presentation	Finalized Project Report

SDD: System Design Document

U.O: User Equilibrium for a Transport Network

S.O: System Optimum for a Transport Network

RAD: Requirements Analysis Document

TNA: Traffic Network Analyzer

RC: Release Candidate

1D.About this document

The structure of this document is as follows. Section 2 describes the project analysis phase from the point of view of the coordinators. Section 3 describes the process of designing and requirements gathering and specifications. Section 4 the implementation related aspects of the software will be explained followed by testing activates in section 5. Section 6 includes the conclusions and recommendations formed and gathered during the project.

2.Analysis

This chapter describes the analysis phase of the project. The analysis phase was divided into two categories of Theoretical and Technical analysis, which were carried out parallel to each other. In theoretical analysis phase various network optimization algorithms were studied whereas in technical analysis phase the software engineering aspects of TNA program was studied.

2A.Theoretical Analysis

We started this phase by first making a list of available algorithms which were already in use for analyzing the performance of the network. After trying various networks algorithm was completed it was decided by Mr. Melissen that *Minimum Delay Flow Deviation Algorithm* will be used to find the optimum flow of various commodities through the network. In this algorithm the flow values of various commodities, their source, sink and the delay function of the network edges across which the commodities must flow would be given in advance. Then using this algorithm an optimal flow route will be planned for the commodities. With optimal flow route we mean a route in which the total travel time delay for all commodities is minimal from the network point of view.

After deciding on which algorithm to be implemented we moved on to see what will be the required functional and non functional requirements from a program that runs this algorithm. In other words what are the functional requirements of the TNA from the point of view of project supervisor? As mentioned above the project was seen by coordinators from two different perspectives (i.e. Software engineering and Optimization technology). Since the network analysis algorithms are mostly computed and executed by mathematicians who have a thorough knowledge of the theory behind the algorithms we spent a considerable amount of time to specify the input output and actual steps need to implement and integrate the algorithm into a java program. That was the reason why we investigated a number of other network related algorithms which lead us to finally decide that one with minimum number of steps and most compact one will be the computational engine of TNA. Since the TNA software from the optimization perspective was to see how does the TNA program performs in terms accuracy, convergence rate and network size. So we finalized the theoretical analysis phase by manually solving an example using Minimum Delay algorithm.

Below is the *Minimum Delay Flow Deviation Algorithm:*

The Minimum Delay Flow Deviation Algorithm

Step 1: Given the feasible flow pattern $\hat{\mathbf{x}}^{\nu}$, determine the partial derivatives $\eta_{ij}^{\nu} = \frac{\partial c_{ij}(x_{ij})}{\partial x_{ij}}$ evaluated at $\hat{x}_{ij}^{\nu}$.

Step 2: For each $k = 1, ..., p$, determine the shortest path from n_{s^k} to n_{t^k} based on the arc values η_{ij}^{ν}. Send $\tilde{f}_k$ units on the shortest path from n_{s^k} to n_{t^k}, $k = 1, ..., p$. The resulting flow pattern is denoted by $\tilde{\mathbf{x}}^{\nu}$.

Step 3: Perform an interpolation search to solve the single-variable problem
Min $z((1-\alpha)\hat{\mathbf{x}}^{\nu} + \alpha\tilde{\mathbf{x}}^{\nu})$, s.t. $\alpha \in [0; 1]$.
Denote the result by $\bar{\alpha}$ and compute $\hat{\mathbf{x}}^{\nu+1} = (1-\bar{\alpha})\hat{\mathbf{x}}^{\nu} + \bar{\alpha}\tilde{\mathbf{x}}^{\nu}$.

Step 4: Is the chosen stop criterion satisfied?
If yes: Stop; a near-optimal solution has been found.
If no: Set $\nu := \nu + 1$ and go to Step 1.

Figure 1:Minimum Delay Flow Deviation Algorithm

2B.Technical Analysis:

TNA was the first such program without any predecessor therefore the set of initial requirements was very vague. The first logical first step was to refine and put in concrete terms the functional requirements. The functional requirements were achieved through discussion and study of various algorithms, so that how far did these algorithms met the originally agreed upon objective of the project

In the initial analyzing phase of the project we assumed that Java and MATLAB would be used in combination for implementation of software program. But after some consultation with some java experts it was very soon discovered that there was no standard driver or interface between the two programming languages. So we either had to make some sort of Matlab java driver which will connect the two languages or use a third programming language which already had a Matlab driver like C. But then in using C we had to compromise the mobility of the software program. After balancing out the advantages and disadvantages of using a combination of Java and Mat lab for implementation, it was opted to skip the use of Matlab for computation and just use Java as the sole programming language. The next phase in the technical analysis was to see how far java supports mathematical and numerical operations like integration differentiation, partial integrations and differentiations etc because these operations form the backbone of the any algorithm that needed to be implemented. As it turned out the open source package of java Science was an ideal tool to use for computations purposes in java.

Then the focused was put on input output mechanism through the GUI. The design of GUI started with a paper prototype which was presented to Mr. Sodyer. Once the main

components of GUI was agreed upon a GUI demo was created using java.Swing and again was presented to project coordinator for receiving their approval. Figure below show the very first paper prototype of the GUI on the left and the first demo of TNA on the right.

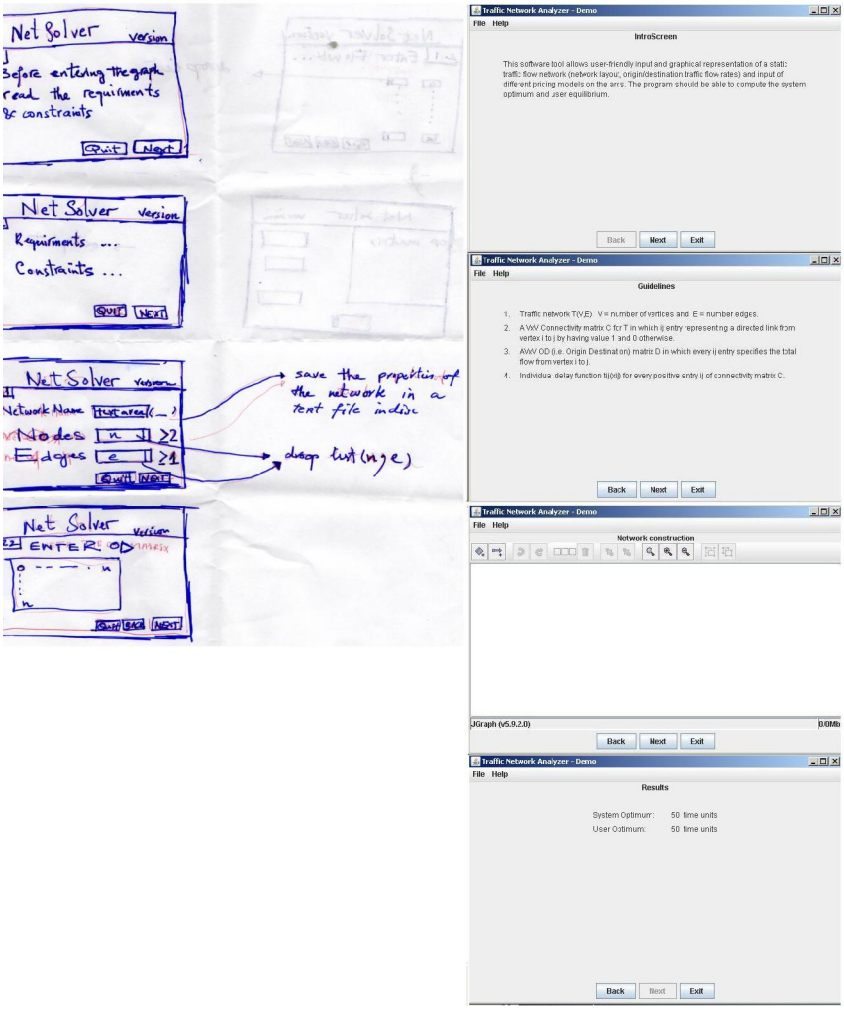


Figure 2

Once the GUI or Black Box model of the system was decided upon, or more specifically said visualized we started to work on the actual implementation or white box model of the system.

We used two open source frameworks that supported our work namely JGraph and Java Science packages. But these frameworks were in the early stages of their development so

they were first thoroughly investigated tested before learning how to use them. Then we tried to understand how to use the numerical operations that the java science package supported.

3.Design

Generating RAD (Requirements Analysis Document) which captured both dynamic and static behavior of the system (Activity Diagram, Class Diagram Architectural Design etc) was done in the design phase of the project. Since the program was implemented using Java programming language thus UML diagrams was the logical choice during the design phase. During the design phase we

The design phase began with the creating UML Activity Diagram (Program Flow Graph) which gave an overview of how the program should be working specially from the user point of view. Once that was achieved I moved on to list the possible classes which will be required to realize TNA software and also the required classes from science and jGraph packages were stripped of to be used.

3A.Activity Diagram:

The following figure depicts the Activity Diagram

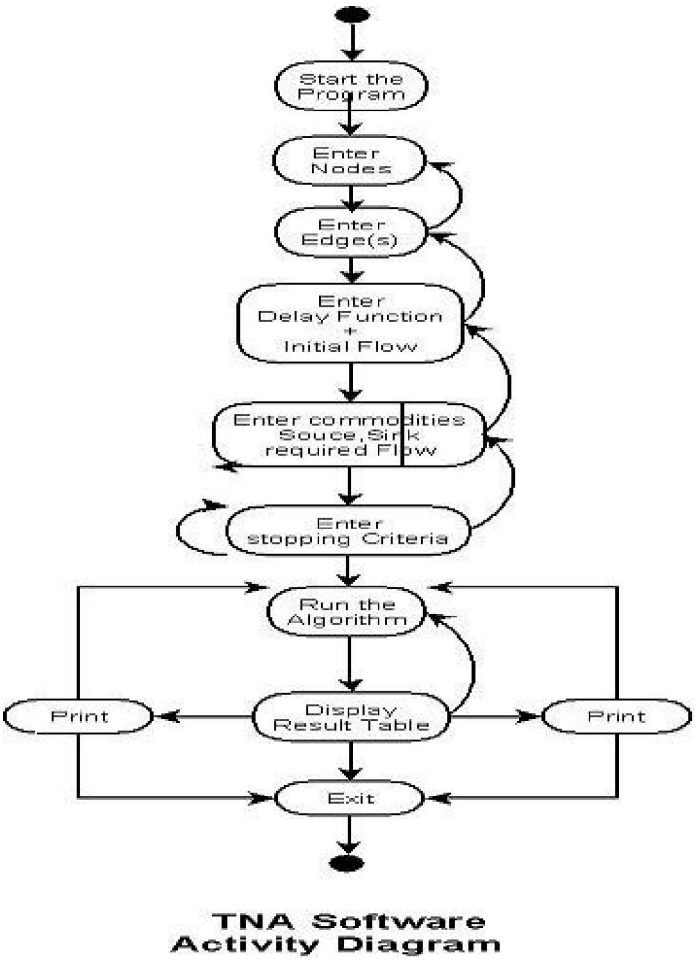


Figure 3:Activity Diagram

3B.Class Diagram:

The first and most important class that was designed was the class which would contain the actual implementation of the Minimum Flow Deviation Algorithm. Once the properties and attributes of this class were specified I decided to analyze and design the input and output adopters for this class. Thus keeping this class as the core the following class diagram model was created.

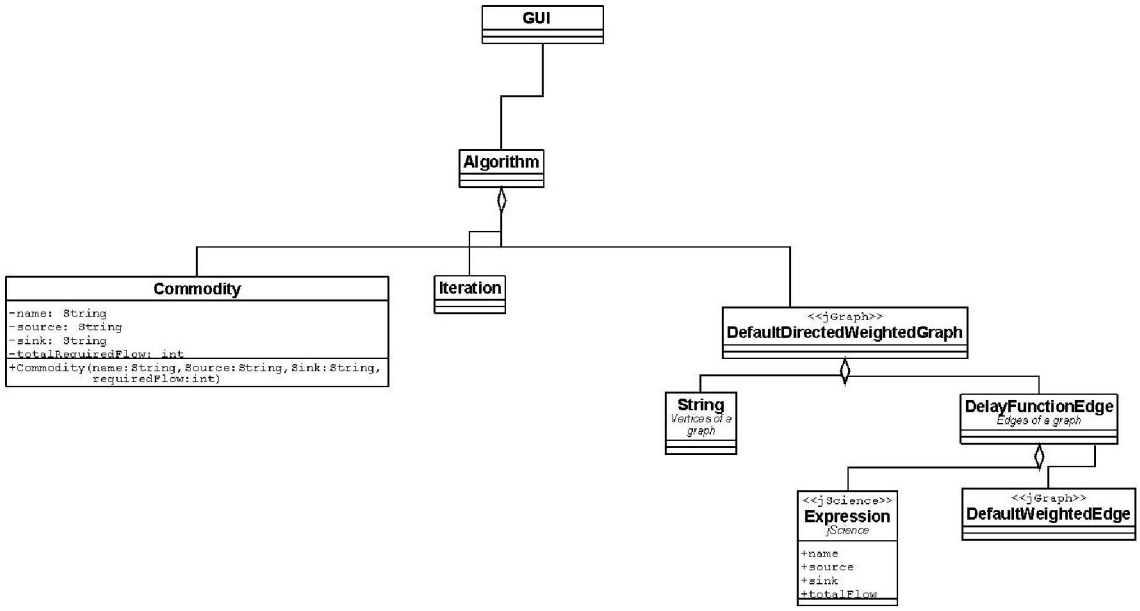


Figure 4:Class Diagram

3C.Architectural Model

Once the class model was obtained I moved on to create an architectural model of the software which would provide me an overview of how and what various external components of software are

how are they interrelated related. The following diagram shows the Architectural Model of TNA.

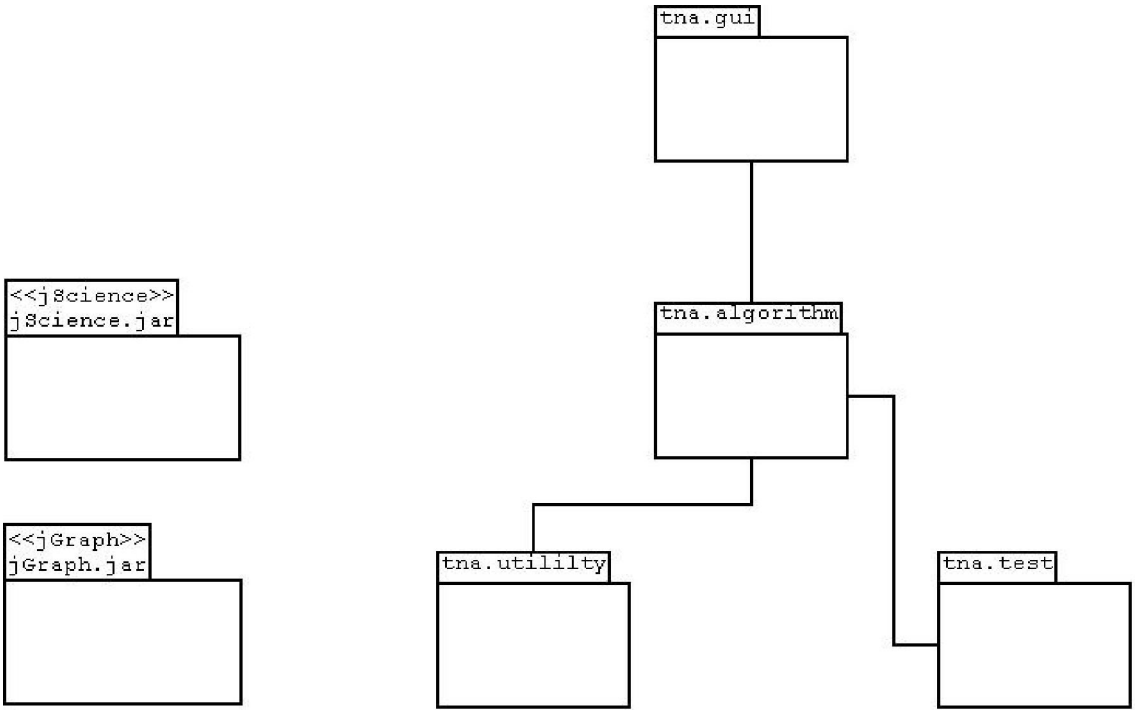


Figure 5:Architectural Model

4.Implementation

This section describes the technology and techniques that were used during the implementation phase of the project. First, the technologies that were used are discussed. Next, the planning is shown. After that, the process that was followed during this phase will be explained and encountered problems will be discussed. Finally, the final design of the system as it was implemented is discussed.

Since almost all of the supporting code for the TNA software project is written in Java, it was a logical choice to implement this project in Java too. The personal experience with Java allowed a quick start-up.
The tools and libraries used in developing the final product are the following:

- Java JDK 1.6.0 The Java development kit
- Eclipse SDK 3.2RC7 A free IDE, supporting Java Programming.
- jUnit 4.1 A Unit testing framework for Java 5.0
- Open Office version 2.3.1, used for generating reports.

Besides the above mentioned tools the standard libraries of java was used for creating GUI of TNA.

The first implementation phase took longer actually longer than expected. This was mainly caused due to inadequate link between the software engineering and mathematical aspects of the project. This was fixed after the some research and analysis as mentioned in the previous sections.
Alongside the implementation of the main program functionality, basic developer testing was conducted using the already solved network problems. In addition to that, unit tests were written in parallel to the development of the main code, using JUnit. For more information about the conducted testing, test plan document should be seen.
The first stage of actual implementation began by making a demo of the desired GUI for the TNA as shown in figure 1. Through this demo GUI it was decided on how the program flow should be. Then a specific example (provided by Mr. Melissen) was picked up and hardcoded (using extreme Programming principles) ,step by step to see what are the exact requirements in terms of available functionality from jScience and jGraph packages. This was actually the most complicated of implementation. Once an example was hardcoded and provided the same results as was the manual calculation the third phase of implementation began. In this phase the actual TNS software was programmed using the example as the point of reference. Along with coding also some limited crucial test classes were built using jUnit. Finally TNA version 1.1 was created as a fat java archive file(*fat jar*). This file is ready to be run on any computer without the requirement of java to be pre-installed. This makes the product 100% platform independent.

5.Testing

The testing strategy for TNA software was categorized into the following three main categories.

1. White box testing
2. Black box testing
3. User Acceptance Testing

5A.White Box Testing

White Box testing was done parallel with implementation and it included the following approaches:

1. Unit Testing
2. GUI testing

The next paragraph explains how the Black box testing strategy was used for TNA. The last paragraph explains how the White Box testing was achieved.

5B.Black Box Testing

According to Black box testing which is also called functional testing the TNA system is viewed as a black box with only input and output possibilities. The Black box testing strategy uses Functional requirements and program specification as its starting point for creating test cases. In order to perform systematic black box testing of TNA software we carried out the following steps.

1. Indentify Independently testable feature of Functional Requirements
2. Identify classes of representative values that must tested for every independently testable feature.
3. Create a model (state chart diagram) for every independently testable feature.
4. Using the representative values (identified in step 2) and model (created in step 3) generate test case specification for every independently testable feature.
5. Finally use the test case specification to create actual test cases for performing black box testing on the system.

.

5C.User Acceptance Testing:

This test was done after the previously mentioned tests has been successfully conducted and was used as a way to see how the system is used in real life. Someone who uses the system in actual situation will be used to running and use the software and the level to which the software results deviates from the one provide by manual calculations.

6.Summary and Recommendations

This chapter summarizes this document. It starts with the list of the functionalities of TNA and mentioning the some main deliverables of the project and is followed by suggestions on extension possibilities for the software piece by other programmers.

What is product goal?

Below are the main functionalities of TNA software:

- Enter properties of the nodes and edges of a given network.
- Enter different kind of commodities that is needed to flow through the network by specifying the total inflow, source and sink.
- Calculate the total delay using the flow deviation algorithm.

TNA software tool is the final result of this project. It allows user-friendly input of network topology and specification. The program is able to compute the minimum delay of the travel time for the network users. The design and implementation of software is done using standard object oriented software engineering principles and technology.
What is delivered?

The final application is delivered as an executable and extractable java archive file (jar) to Project coordinators. Also a RAD (Requirements Analysis Document), javadoc of the source code, a read me file and a test plan document for the TNA software is provided to the instructors.
The TNA application and documents related to the project meet most of the originally agreed upon goals of the project. All of the functional requirements documented in the requirements analysis documents have been implemented. Thus it can be conclude that Traffic Network Analyzer project has been successful.

For further improvement and extension of the TNA program some extra functionality can be attached like the program should be able to interact with network system which Recommendation1

Another major improvement can be brought about by allowing TNA to compute the required minimum delay values for dynamic networks where the network in and out flows change dynamically. Recommendation2

Finally the implemented program can be stripped of using reverse engineering techniques such that input output mechanism becomes absolutely independent of the algorithm it is working with and this input outputs mechanism is used as a framework for other graph and network analysis algorithm. Using this framework the future network algorithms

programmers do not have to bother about different aspects of input and output or input and output adopters. Recommendation3

Final Personal Remarks

During the course of this project I achieved and received some very valuable guidance and experiences from both the coordinators and my working experience. This project was my first experience in building fully operable software from the scratch.

During the design phase I have realized that actually this was the most crucial phase of the project. It is very crucial to generate or design valid models which are implantable but at the same time programmer independent. Such that any programmer can use this model to extend modify or re-implement the TNA software.

As mentioned above building software from scratch requires an intensive requirements gathering and experience which I did not had in abundance. These factor leads to major loss of momentum in the beginning of the project. Also not to mention since I was working alone I was lacking enough critical analysis and discussion from and with another professional about project management's aspects. Finally I m glad to conclude that the agreed upon objectives of the project was achieved in an acceptable manner.

Appendixes

Appendix A: Plan of Approach version 1.5

Plan of Approach

Bachelor Project IN3700

Khaled Tokhi

Student number: 1111647

Date: October 9 2006

Status: Approved

Version: 1.5

Table of Contents

1 Introduction

This document serves as a plan of approach for Bachelor Project IN3700, which is carried out under, mixed guidance and supervision of Dr. J.B.M. Melissen from Information Systems and Algorithms department and Ir. B.R. Sodoyer from Information Systems Design Department.
The main goal of this document is to specify the following points:

- Project Objective
- Project Domain
- Project Deliverables
- Time Schedule of the Project

The structure of this document is as follows. Section 2 describes various aspects of the project from the point of view of the project initiator. Section 3 describes and specifies the main deliverables and time schedule of the project form the point of view of both project initiator and supervisors.

2 Project Description

2a. Project Domain: The domain in which this project is carried out is a combination of optimization methods and software engineering principles.

2b. Project Objective: Main objective of project is to compute/analyze the effect of taxing a subset of roads (links) on user equilibrium and system optimum of a static traffic network.

2c. Project Assignment: Project assignment can be defined as the study of different optimization algorithms for computing user equilibrium and system optimum in a static traffic network before and after taxing some links in the networks. Then implementing these methods in a computer program using Java and MATLAB as programming languages while applying standard software engineering techniques and principles.

2d Deliverables: The main deliverables of this project are the following:

1. **Plan of Approach**
2. **Computer Program**
3. **Project Report**

2e. General Agreements: The software tool, which should be developed, should allow user-friendly input of network specification and different pricing models. The program should be able to compute the system optimum and user equilibrium with and without taxing and compare them quantitatively and qualitatively. The design and implementation should be done using standard software engineering tools and techniques.

3 Project Organization

This section describes responsibilities of people involved in the project, main deliverable of the project in general, hardware and software specification and the time schedules of the project.

3a. People Involved

The people who are involved in this project are

1. Dr. J.B.M. Melissen Project Initiator/ Supervisor

2. Ir. B.R. Sodoyer Supervisor

3. Khaled Tokhi Student

Since this project was not done as a standard software engineering project in which a group supposes to carry it out in a company/organization therefore the role of supervision is split between Mr. Melissen who will supervise in Optimization part and Mr. Sodyer who will supervise the Software Engineering aspects.

3b. Deliverables

The main deliverables as mentioned in Section 2d are the followings:

1. **Plan of Approach:** This is a global planning for the project at the start of the project.
2. **RAD (Requirements Analysis Document)...**
3. **Computer Program:** The software tool TNA (Traffic Network Analyzer), which compute and visualize the effects of pricing strategies in a given network plus a user guide. It will be produced in 2 version beta and final**.** The end product will be a java….The software
4. **Project Report:** This is a written report containing description of the literature study, a description of all software engineering aspects, and the results of tests on some practical situations.

3c. Hardware/Software Specification

Object oriented programming language Java (JDK 5.0) in combination with MATLAB mathematical programming language is used for implementation/computational purposes.

NetBeans IDE 5 would be used to document, run, clean, test, and debug the java application.

No specialized hardware is used except for normal desktop workstations running in Windows environment in the faculty labs.

3d. Time Schedule

As result of unexpected regulation for some course the project need to be stopped temporarily until February. The new schedule for the rest of the project is represented in the following table:

Week	Project Week	Project Phase	Deliverables (Date)

45 (06.11.2006)	5	Software Design	1.SDD 1.1 (7.11.2006) 2.Core Algorithm for calculating U.O and S.E (8.11.2006) 3.RAD version 1.2(9.11.2006) 4.Mid Term Evaluation Report (10.11.2006)
		November	No Activity
		December	No Activity
		January	No Activity
6 (05.02.2007)	6	Implementation	1.Plan of Approach version 1.5 2.Table of Content of Project Report Test Plan(5.02.2007)
7	7	Implementation	
8	8	Debugging/Testing of code	Beta Version of TNA
9	9	Testing of Program	RC1 of TNA
10	10	Project Report	Final Version
11	11	Prepare A Draft Report of Project	First version of Project Report
12	12	Prepare Project Presentation	Finalized Project Report

SDD: System Design Document

U.O: User Equilibrium for a Transport Network

S.O: System Optimum for a Transport Network

RAD: Requirements Analysis Document

TNA: Traffic Network Analyzer
RC: Release Candidate

Appendix B: Requirements Analysis Document:

Traffic Network Analyzer

Bachelor Project IN3700

Requirements Analysis Document

Khaled Tokhi

Student Number : 1111647

Version : 1.1

Contents

1 Introduction

This document serves as a RAD Requirements Analysis Document for software system TNA Traffic Network Analyzer, which is done in partial fulfilment of Bachelor Project IN370.

2 Current System

One of the major problem in the traffic networks is the selfish behaviour of commuters who use the network or in other words every commuter tries to reach his/her destination in the shortest time without regard to the other commuters. This behaviour generally leads to an overload in some roads where the number of commuter exceeds the road capacity and thus causes traffic jams in which everyone in the road spend more time. Therefore in order to balance the effect of selfish behaviour of commuters traffic authorities need to take some measures in which one of them is by taxing some roads they will be

3 Proposed System

Traffic authorities want to add some sort of tax to the roads which get jammed frequently but in order to do so they need to know the difference between system optimum and user optimum. If the difference is too big they need to tax them. This software helps them in calculating user optimum and system optimum.

3.1 Overview

The software tool, which should be developed, should allow user-friendly input of network topology and specification and different pricing models. The program should be able to compute the system optimum and user equilibrium. The design and implementation is done using standard software engineering tools and techniques

3.2 Functional Requirements

TNA system has the following as its primary user:

Traffic Authorities are those people who analyze the roads in a traffic network. They analyze effects of traffic overflow in certain roads (paths) based in their observation they decide on whether to tax some roads which frequently gets overflowed and cause extra travel time for the commuters. Traffic authorities expect the following functionalities from the system:

3.2.1 Enter topography of network in the system in the from of an OD matrix
3.2.2 Enter properties of the nodes and edges
3.2.3 Get SE (System Equilibrium)
3.2.4 Get UE (User Optimum)

3.3 Non functional requirements

3.3.1 User interface and human factors

GUI's of the TNA system should be designed in such a way so that every user of the system should feel comfortable with it, without having a great deal of knowledge about computers.

3.3.2 **Documentation**

A user manual is required. Secondly, the software must be documented very well to make maintenance easy for developers who want to extend it and implement or integrate related code to it.

3.3.3 **Hardware considerations:**

TNA software should be able to run on today's personal home computers.

3.3.4 **Performance characteristics:**

TNA should be able to provide an output for any input and otherwise provide an error message.

3.3.5 **System interfacing**

This system do not have to interface with other systems

3.3.6

System modifications.

None

3.3.7

System modifications.

None

3.3.8

Physical environment

None.

Security issues

None

Resources and management issues

None.

3.4 **Constraints ("Pseudo requirements")**

Java should be used as programming language for building the system and MATLAB mathematical programming language as an option for difficult computations

3.5 **System models**

3.5.1 **Use Cases**

Use Case (see FR: 3.2.1.1)

Name:	Enter the network in the system
Actors:	Traffic network authorities
Entrance condition:	User should know the network's topography.
Flow of events:	

1. Actor starts the system.
2. Actor enters properties of the nodes and vertices.

Exit condition: Actor gets all the entered information represented by TNA system.

Use Case (see FR: 3.2.2)
Name: Enter Network Properties.

Actors: Traffic network authorities

Flow of events:

1. Actor enters the network in the system
2. Actors enters the properties of the nodes and edges
3. Actor clicks on "(Start)" button of GUI

Exit condition: The actor goes to stage of being ready to run the algorithm.

Exit condition: Actor gets all the entered information represented by TNA system.

Use Case (see FR: 3.2.3)
Name: Get Minimum Delay Time of a Traffic network.

Actors: Traffic network authorities

Flow of events:

1. Actor enters the network in the system
2. Actors enters the properties of the nodes and edges
3. Actor clicks on "(Start)" button of GUI

Exit condition: The actor has the computed minimum delay of the network on his/her screen.

Exit condition: Actor gets all the entered information represented by TNA system.

Use Case (see FR: 3.2.1.4)
Name: Save a Traffic network.

Actors: Traffic network authorities

Flow of events:

4. Actor enters the network in the system
5. Actors enters the properties of the nodes and edges
6. Actor clicks on save button of GUI

Exit condition: The actor has the network save in the system.

Use Case (see FR: 3.2.1.5)

Name: Print the result table.

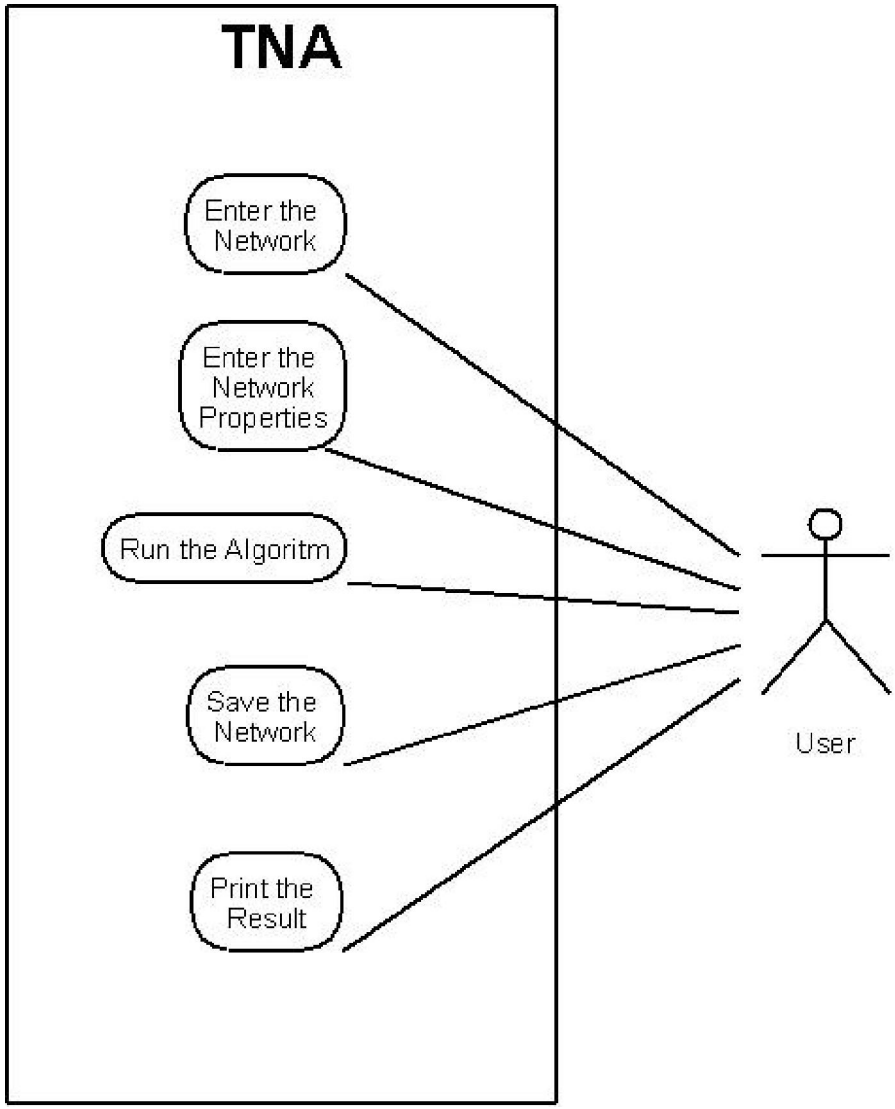
Actors: Traffic network authorities

Flow of events:

1. Actor enters the network in the system
2. Actors enters the properties of the nodes and edges
3. Actor clicks on print button of GUI

Exit condition: The actor has the network printed to a file or printer.

Use case models

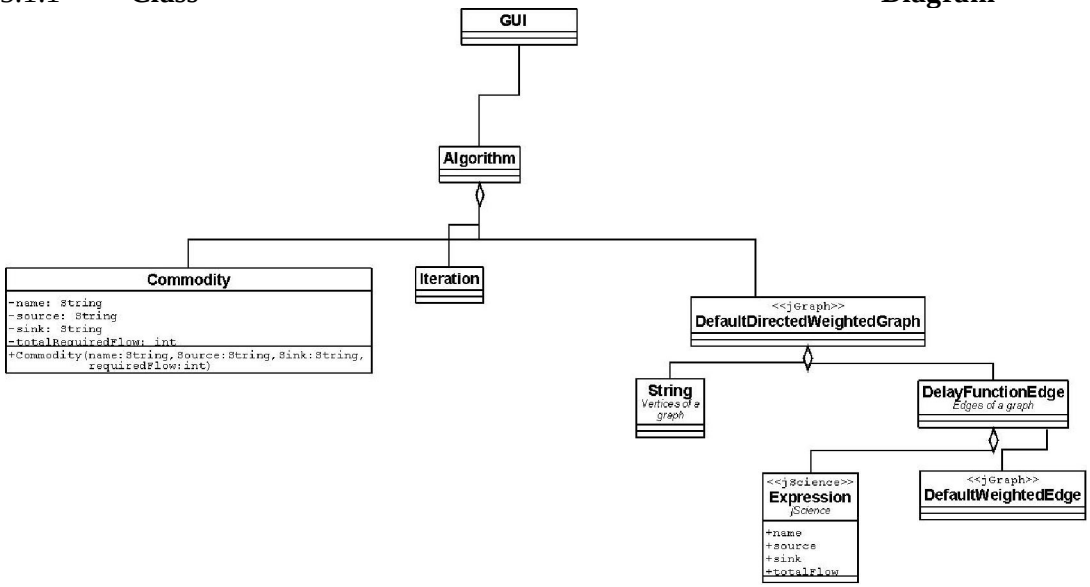


Object Models

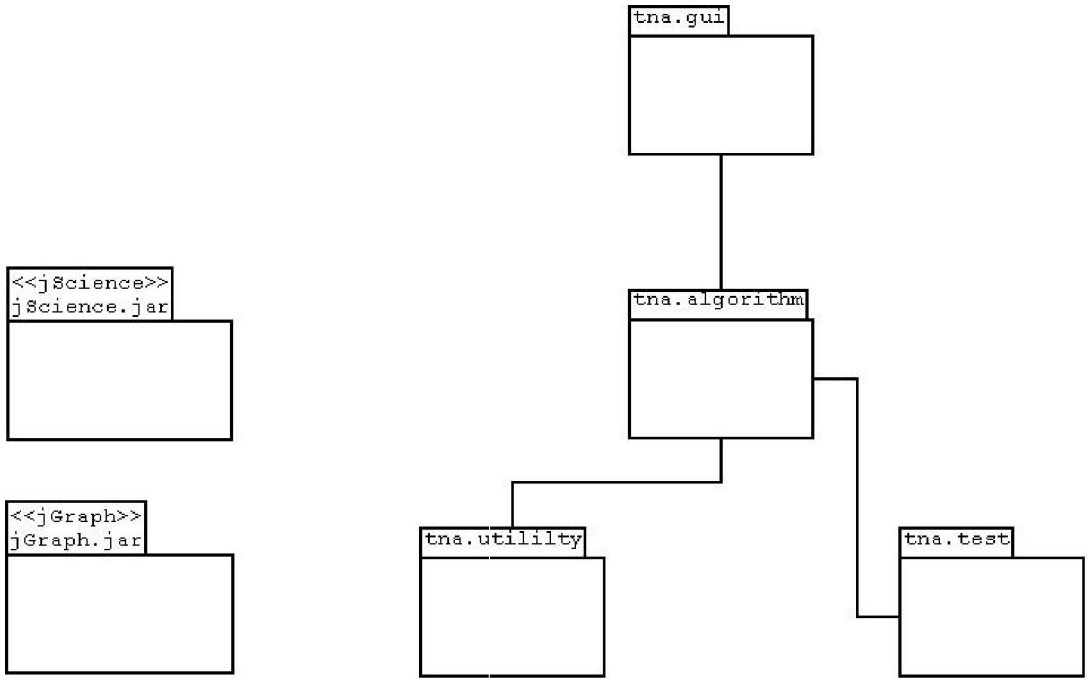
Data Dictionary

Commodity :
Source :
Sink
Delay Function :

3.5.1.1 **Class Diagram**



3.5.2 **Architectural design**



Appendix D: Test Plan of TNA

TNA Test Plan

Khaled Tokhi
Student Number : 1111647

Version : 1.2

Status : Verified

Introduction:

This document serves as a Test Plan for the TNA (Traffic Network Analyzer) software which will be developed as a partial requirement for bachelor project of Technical informatics IN3700.

1. Unit Testing:

This testing method is used as a guard against bad coding and is created during the implementation phase using JUnit 1.4(A unit testing framework for Java 5.0).

Unit testing phase will be conducted parallel with implementation phase.

2. GUI testing:

Because there is no mechanism for creating automated tests for GUI the functionality of GUI will be tested manually in an interactive fashion. It will be done by testing every component of the GUI: The tables below will indicate which components of GUI are functioning properly and which not. The result in table 1 is not based on the engine behind the GUI but just as a sort of demo that the flow of events is running properly. Table 2 will describe the function while actually using the engine and providing some input, see Appendix 1A and 1B.

Table 1. GUI Testing table without Data

Component Name	Functionality	Expected Action	Actual Action	Acc	Not-Acc	Partially-Acc
Start Button	Go to Graph	Graph Panel Appears	Graph Panel Appears	Yes	No	No
Add node	Node appears	Node is created	Node is created			
Add edge	Edge appears	Edge is created	Edge is created			
Connect nodes	Two nodes are connected	Edge is created				
Enter Edge Properties	Delay_functi+ initital flow is added	Data is stored	Data is stored	Yes	No	No
Add Commodity	Add a/multiple commodities	Commodity for a source & destination is added	Commodity for a source destination is added	YES	No	No
Enter the stopping Criteria	A double number is entered in the text field					

Run Button	Show the result table	Result table appears	Result table appears	YES	No	No
Re-Run Button	Show the result table	Result table appears	Result table appears	YES	No	No
Exit Button	Exit the Program	Exit the Program	Exit the Program	Yes	No	No

Table 2. GUI Testing table with data on appendix 1A

Component Name	Functionality	Expected Action	Actual Action	Acc	Not-Acc	Partially-Acc

Table 3.GUI Testing table with data on appendix 1B

Component Name	Functionality	Expected Res	Actual Result	Acc	Not-Acc	Partially-Acc

Acc=Acceptable

3. Functionality Testing:

These tests will check the software as whole so the measure of testing will be how much of the functional requirements are satisfied after the software is put together will be checked using this

technique. In table 4 the functional requirements of the system are picked from RAD document and checked using data in Appendix 1A and 1B.

Table 4. Functional Testing Table with Data in Appendix 1A

FR number	Description	Data	Restriction	Actual-Result	Expected Result
1	Enter network specification T(N,V)				
2	Enter the number of commodities				
3	Enter the stopping criteria				
3	Get Minimum Delay				

Table 5. Functional Testing Table with Data in Appendix 1B

FR number	Description	Data	Restriction	Actual-Result	Expected Result
1	Enter network specification T(N,V),delay function, initial flow				
2	Enter the number of commodities				
3	Enter the stopping criteria				
4	Get Minimum Delay				

A = Acceptable

NA = Not Acceptable

PA = Partially Acceptable

4. Boundary Testing:

This test is conducted by using the upper extreme situations and see how the software behaves. The data in this kind of testing is usually fictitious and must be delivered by the project coordinator Mr. Melissen.

4. User Acceptance Testing:

This test will be done after the previously mentioned tests are successfully conducted and is used as a way to see how the system is used in real life someone who will use the system in actual situation will be used to run and use the software and the level to which the software. This mechanism for this test will be created at the end of implementation phase. The data at Appendix 2 will be used as a input with an already manually calculated output. Here the user will get an idea of how much the actual result deviates from the one provide by the TNA software. Also performance of the TNA software will be made visible to the user in terms of speed and accuracy.

Appendix 1A: Testing Data

1. Network name : 1A

2. Number of nodes : 4

3. Number of edges : 5

4. Total Required Flow : 6 units

5. Connectivity Matrix :

	1	2	3	4
1	0	1	1	0
2	-1	0	1	1
3	-1	-1	0	1
4	0	0	-1	-1

6. Origin Destination Matrix :

	1	2	3	4
1	0	0	0	6
2	0	0	0	0
3	0	0	0	0
4				

7. Delay Function Matrix :

	1	2	3	4
1	-	8 + (2x*12*)	33 +(x*13*)	-
2	-	-	13 + (3x*13*)	8 +2(x*24*)
3	-	-	-	33 +(x*34*)
4	-	-	-	-

8. Minimum Delay z :

9. Minimum Delay z :

Appendix 1B: Testing Data

1. Network name : 1B

2. Number of nodes : 4

3. Number of edges : 5

4. Total Required Flow : 5 units

5. Connectivity Matrix :

	1	2	3	4
1	0	0	1	1
2	0	0	1	1
3	-1	-1	0	1
4	-1	-1	-1	0

6. Origin Destination Matrix :

	1	2	3	4
1	0	0	0	5
2	0	0	0	5
3	0	0	0	0
4	0	0	0	0

7. Delay Function Matrix :

	1	2	3	4
1	-1	-1	0	5 + (x*14*)
2	-1	-1	0	7 + (x*24*)
3	-1	-1	-1	9
4	-1	-1	-1	-1

8. Minimum Delay z :

9. Minimum Delay z :

:

Appendix 2A: Testing Data

1. Network name : 2A

2. Number of nodes : 4

3. Number of edges : 5

4. Total Required Flow : 5 units

5. Connectivity Matrix :

	1	2	3	4
1	0	0	1	1
2	0	0	1	1
3	-1	-1	0	1
4	-1	-1	-1	0

6. Origin Destination Matrix :

	1	2	3	4
1	0	0	0	5
2	0	0	0	5
3	0	0	0	0
4	0	0	0	0

7. Delay Function Matrix :

	1	2	3	4
1	-1	-1	0	5 + (x*14*)
2	-1	-1	0	7 + (x*24*)
3	-1	-1	-1	9
4	-1	-1	-1	-1

8. Minimum Delay z :

9. Minimum Delay z :

:

References:

1. B.Bruegge & A.H. Dutoit, *Object-Oriented Software Engineering: Conquering Complex and Changing Systems*, vol. 1, Prentice Hall 2000.

2. G.Booch,J.Rumbaugh & I.Jacobson, *The Unified Modeling Language User Guide*, vol. 1, Addison-Wesly 1999.

3. R.V. Binder, Testing Object-Oriented Systems: Models,Patterns and Tools, vol. 1, Addison-Wesly 2000.

4. http://java.sun.com/

5. http:// jgraph.org/

6. http://jscience.org