The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)___________________
Summary

This project presents a prototype to design a schematic bus route. The prototype will be capable of utilising a structured map as its input, and outputs the bus routes into a valid structured map. However, the project does not cover the subject on automated construction of such structured map.

By performing evaluation on existing structured maps, this project aims to produce its own artificial structured map to be used by the prototype. Finally, a thorough evaluation on the prototype is conducted to assess its efficiency. Future expansion on the prototype is highly suggested.
Acknowledgements

Firstly, I would like to thank Dr Raymond Kwan for his advice and hints on how this project should be carried out. I would also like to thank Dr David Duke for his comments on the mid-project report. Although, in my opinion, they are a little bit harsh but nevertheless very accurate. The comments had given me a lot of idea and allowed me to make progress.

Thank you to everybody who had helped me with my research along the way. Particularly to Deborah Allon and Anna Clough for their assistance in Ordnance Survey products, to Dr Bill Lythgoe and Dr David Milne of ITS for their help on SATURN.

A very big thank you to all member of ISS staffs, especially to Naush, Peter, Dureid (who prefers to be called Dr El-Moghraby), Robert, Ed, Mazin and other out-of-hours supervisors. Working with all of you have been a great pleasure. I am “almost” looking forward to every shift, “almost”.

A special thank you for everyone at The North East Leeds City Learning Centre, in particular to Peter Fletcher, Claire Easton, Sue Cassidy, Aisling McGibbon and Anna Stevens. They had been exceptionally kind to me during my industrial placement. Working in NECLC had been the most memorable event in my life accompanied by Peter’s jokes that I never get, Claire’s sweet personality, Sue’s liveliness, Aisling’s kindness and Anna’s practical jokes. I could not wish for a better working environment.

Thank you to all member of SoC staffs. Getting educated in England was a fantasy that I could have never dared to dream about before. A personal thank to Dr Stuart Robert who recommended me to NECLC, Tony Jenkins with his “unique” sense of humour and Dr Pete Jimack who is an excellent tutor.

I would like express my gratitude to my aunt who have been supporting me morally and financially all my life. Thank you for giving me the chance to experience the best education. I truly hope that I could repay you someday. A special thank you to my mother, who has always love me.

A very special thank you to my best friend, Awi who has always stood by my side. Always ...

Finally, to my companion .....
Irawaty, I owe you more than I could ever express
# Contents

## 1 Introduction

1.1 Overview ................................................................. 1

1.2 Brief History of Digital Cartography ................................. 2

1.3 Project Management .................................................... 2

1.3.1 Aims and Objectives ............................................... 2

1.3.1.1 Project Aims .................................................. 2

1.3.1.2 Project Objectives .......................................... 3

1.3.1.3 Minimum Requirements .................................... 3

1.3.2 Schedule .............................................................. 5

1.3.2.1 Project Schedule ............................................. 5

1.3.2.2 Progress vs. Schedule ...................................... 6

## 2 Background Research

2.1 Background to the problem ............................................. 7

2.2 Type of Map Design ................................................... 9

2.2.1 Topological Map .................................................. 9

2.2.2 Schematic Map ................................................... 10

2.3 Structured Maps ...................................................... 12

2.3.1 Meridian 2 and Strategi ......................................... 13

2.3.2 SATURN .......................................................... 15

2.4 Methodology ........................................................... 17
5.2.1 Parsing the Structured Map .............................................. 45
5.2.2 Road Selection Process ................................................ 46
5.3 Proposed Extensions ....................................................... 47

Bibliography .......................................................................... 50

A Personal Reflection ........................................................... 52
Chapter 1

Introduction

1.1 Overview

This project aims to produce a prototype application that would design and edit a schematic graphical bus route map interactively. This prototype would display a map by importing data from digital map data files or structured map (the latter term will be used to describe this type of map throughout this report).

A structured map is a file that contains information to define a map precisely by using descriptive data. This type of map is more practical to be used by the prototype as opposed to normal image-based maps. The prototype should also be capable of displaying, editing, exporting and importing bus routes interactively.
1.2 Brief History of Digital Cartography

In the early development of computer-aided cartography, the whole process of mapping is not fully automated. The input to the system had to be carried out manually, and only the display transformation was automated. One of these early systems is SYMAP or Synagraphic Mapping System, developed by the Harvard Laboratory for Computer Graphics and Spatial Analysis [1]. SYMAP only accepts punched card as input and restricts its output to a line printer.

However, the rapid development of processing power has changed the world of cartography immensely. Nowadays, most modern mapping programs are fully automated from input, to data processing and complex transformation, to output.

Many different terms are used to describe this development such as digital cartography, computer-assisted cartography, computer-aided cartography and automated cartography. The term digital cartography will be used throughout this report.

Digital Cartography is a broad field, and this project will only focus on how spatial data could be represented in a structured map. It will not cover any information regarding automated reconstruction of a structured map, nor will it discuss three-dimensional cartography.

1.3 Project Management

1.3.1 Aims and Objectives

1.3.1.1 Project Aims

The overall aim of the project is to develop prototype software that would represent bus route maps schematically.

The prototype would use a structured map as its input. It should also be capable of handling user
interaction with the system interface to create bus routes and bus stops, which are overlaid on top of the roads. The superimposed routes, as well as the roads of the map could be switched on and off to provide user with a more transparent view of the map. Finally, all the bus routes and bus stops, that were created, could be exported back into an image file or even a structured map. This “Route Structured Map” could then be used to import the route back into the application with ease.

1.3.1.2 Project Objectives

The objectives of this project are to:

- Research on available structured maps such as Ordnance Survey maps and Saturn structured map.
- Analyse and in-depth understanding on how data can be represented in different structured map.
- Design a simplified artificial structured map that is suitable to be used as the basis for the Schematic Bus Route Map Editor.
- Design and build the skeleton of the Schematic Bus Route Map Editor.
- Implement the structured map into the prototype.
- Add functionality to the prototype so it could directly interact with the loaded structured map.
- Improve functionality to edit, export and import bus routes.
- Evaluate the prototype and the project as a whole.

1.3.1.3 Minimum Requirements

The minimum requirements have been modified in response to the assessor comments on the mid-project report. The changes have been agreed with the project supervisor and approved by the assessor.

The following are the revised minimum requirements for this project:
• Present a survey on 2 forms of existing structured maps, examining how data are represented in each one of them.

• Design and model how data would be represented in an artificial schematic structured map. This map would later be processed by the application.

• The program should correctly display the map stored within any given structured map, provided that the structured map is in the right format.

• The program should allow user to interact directly with the map to create bus routes.

• Conduct tests on the prototype to evaluate its functional behaviours, and consider possible and desirable extensions to the system.
### 1.3.2 Schedule

#### 1.3.2.1 Project Schedule

<table>
<thead>
<tr>
<th>Start Date</th>
<th>End Date</th>
<th>Milestones</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct 20, 2004</td>
<td>Nov 10, 2004</td>
<td></td>
<td>Research on existing structured maps</td>
</tr>
<tr>
<td>Nov 11, 2004</td>
<td>Nov 25, 2004</td>
<td></td>
<td>Get sample data of structured maps and conduct a research to analyse the data supplied</td>
</tr>
<tr>
<td>Nov 26, 2004</td>
<td>Nov 30, 2004</td>
<td></td>
<td>Acquire a rough idea on the format of the artificial structured map</td>
</tr>
<tr>
<td>Dec 1, 2004</td>
<td>Dec 9, 2004</td>
<td>Mid-Project Report</td>
<td>Write up the mid-project report</td>
</tr>
<tr>
<td>Christmas</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jan 8, 2004</td>
<td>Jan 18, 2004</td>
<td>Basic prototype</td>
<td>Determine the resources that are intended to be used for building a basic prototype for the application</td>
</tr>
<tr>
<td>Jan 19, 2005</td>
<td>Jan 31, 2005</td>
<td>Prototype version 2</td>
<td>Construct an artificial structured map and amend the prototype to accept and handle it properly</td>
</tr>
<tr>
<td>Feb 1, 2005</td>
<td>Feb 20, 2005</td>
<td>Prototype version 3</td>
<td>Investigate and improve the functionality of the prototype</td>
</tr>
<tr>
<td>Feb 21, 2005</td>
<td>Feb 28, 2005</td>
<td>development chapter</td>
<td>Write up the development process</td>
</tr>
<tr>
<td>March 1, 2005</td>
<td>March 20, 2005</td>
<td>Prototype version 4</td>
<td>Evaluate the project and work on the prototype to expands its functionality wherever possible</td>
</tr>
<tr>
<td>Easter</td>
<td>Final Report</td>
<td></td>
<td>Write the final report</td>
</tr>
</tbody>
</table>

Table 1.1: Project Schedule
1.3.2.2  Progress vs. Schedule

The final result of the project has met all the appropriate milestones outlined in the project schedule (Table 1.1). Although at the beginning of the project, the progress nearly came to a halt due to lack of knowledge and material.

Some difficulties also occurred during the expansion of functionality and writing up of the development process in mid February to March. These difficulties are caused by the increasing workloads from other modules. However despite the escalating workload, the final prototype managed to satisfy all the minimum requirements of the project and even exceeded some of them.
Chapter 2

Background Research

2.1 Background to the problem

Many bus companies are adapting Harry Beck’s London Underground Map [2] design to illustrate their bus routes. The figure 2.1 is an example of how the First company displays its Leeds bus routes.

Figure 2.1: First bus routes for Leeds[3].
Although Harry Beck’s design works very well with the London Underground, it is not adequate to be implemented in designing a bus route. This is due to the fact that unlike underground train design which only display its train routes, in bus routes point of view the roads could be as important as the bus routes itself. Therefore the roads needed to be displayed together with the routes.

As an alternative, some bus companies also provide route maps which includes all the roads (See figure 2.2). This type of map presents its bus routes by placing it on top of the motorways.

![Central London bus routes](image)

Figure 2.2: Central London bus routes[4].

As opposed to previous problem, this particular type of map presents too much details. These details might not be relevant to the user and could conceal and obscure the more important bus routes. To make matter worse, the map is not schematic but topologic, which means the map might looks as complex as the real world. Therefore an intermediate design between the two is desirable.

One solution to the problem might be an application that could create schematic bus routes interactively and offers the user to switch between viewing option. This viewing option should allow the user to view either the bus routes or the roads map on its own, as well as the combination of the two.
To offer users a wider range of viewing options and usability, the application should also provide an alternative output other than displaying the bus routes onto a monitor. For instance, exporting the routes into a structured map so that it could be loaded into the application later on, or saving the output into an image file so that it could be printed out for public distribution.

2.2 Type of Map Design

Designing maps for a transport system is an extensively discussed area of Cartography. Different type of maps could be designed differently to suit the application of the map. This is particularly true when it comes to designing train or bus routes. The following sub-section briefly discussed some of the different approaches that could be used.

2.2.1 Topological Map

Topological map is the traditional method in designing a map. A topological map defines the precise physical location of its geographical objects. In other words, each entity of a topological map has to be correctly represented with regards to its size and position. This means every map is a scaled down representation of the real world.

This method is exceedingly useful for viewers because it would help them to accurately discover crucial information about the environment. Examples of such information could be the position of each entity, the size of a region, the scale of the map, the distance between any two entities of the map. It could even show the depth or height of a surface in three-dimensional cartography.

However, with all these vast amount of detail, complexity could be a problem for some applications. The following is an example of why such complexity might be a disadvantage. A typical motorway usually bends and twists forming a curved motorway, and in the real world, many of the curved motorways might intersect with one another. If the map represents hundreds or even thousands of such motorways, then this could cause the map to be unnecessarily complex. Therefore, since a topological map is a
direct representation of the real world, it might be too complex to be used to illustrate a bus route (see figure 2.2).

2.2.2 Schematic Map

A Schematic map, unlike a topological map, simply considers that an unambiguous and clearly illustrated path between 2 nodes is the main concern. This representation would allow viewers to easily distinguish routes and stops without all the irrelevant details that may obscure a topological map.

Figure 2.3: 1932 London Underground Map - pre Harry Beck [5]

Schematic map were pioneered by Harry Beck in 1933 in the design of the London Underground Map [6]. Beck’s map is completely schematic and bears very little or no relationship with the geographical position of each station or the distance between them.

Beck’s London Underground Map was initially rejected by the London Transport due to the fact that it is not geographical. In his map, all routes are illustrated by straight lines. He simplified them even
further by only allowing horizontal, vertical or diagonal lines (45°) (this adjustment is better illustrated by comparing figure 2.4 and figure 2.5). He also expanded the central area where the routes/stations are much more complex and reduced the outer area. All these adjustments have made the complex London Underground map (see figure 2.4) appear much clearer.

Figure 2.4: 1933 London Underground Map - Original Harry Beck design [7]

Harry Beck’s London Underground Map was so successful that it is now adopted in many public transports map in major cities including Paris, New York, Sydney, Washington D.C., St. Petersburg and Montreal.

For essentially the same reasons, the program developed by this project will employ the schematic approach to the map design rather than a traditional topological approach.
2.3 Structured Maps

A structured map uses a set of data to define spatial information. The spatial information of each entity is depicted using descriptive data. Such data has to be machine readable so that a computer system could understand how a map should be correctly represented.

The use of this type of map maybe more suited for Digital Cartography applications when compared with normal image-based maps. Although it is possible for a program to interpret an image-based map (an example might be: extracting roads from satellite images[9]), the process would be much more complex thus require more computational time and memory to do so. In addition, the prime drawback of such process is that it is particularly prone to errors.

Different types of structured map could be obtained from various companies which specialise themselves in Geographic Information System (GIS)[10]. Ordnance Survey is one of the most well known companies in this field. It offers wide variety of structured maps, each are categorised by the scale of
the map, level of details, and terrain information.

The following sub-sections will discuss some of the different approaches used in structured maps to represent an environment.

### 2.3.1 Meridian 2 and Strategi

Meridian 2 and Strategi are Ordnance Survey products that focus on data concerning motorways.

In Meridian 2 and Strategi, geographic features in the environment are denoted by lines and points. Every lines and points must have certain geometric information attributes and may or may not have extra attributes such as name.

Lines are used to represent roads, railways, rivers, coastline and regions such as administrative areas or woodland areas. Points are used to represent road junctions and all intersections between entities of the map. However, points may also exist independently of lines.

![Figure 2.6: Lines and points represent an entity (image taken from [11] p.17).](image)

Figure 2.6 shows that a line entity (could be a road or a railway) in Meridian 2 and Strategi could be made up of several individual lines. Both point A and B are nodes because they both represent intersection between two different entities.

Figure 2.7 shows how an area is represented in Meridian 2 and Strategi. Point A is the centre of the polygon and it may or may not have a definitive name.

Each entity in Meridian 2 or Strategi may have a set of possible attributes such as CN for county name,
JN for junction name, OW for owner of the property or TX for text associated with the entity. Every entity also exists within tiles and could spread out between more than one tiles as shown in figure 2.8. These tiles were produced by referring to The Ordnance Survey National Grid, which divides The United Kingdom into 100 km × 100 km square area. Each tile has a unique two letters identifier and could be divided into smaller tiles of 100 km² or even 10 km².

A unique identifier code is assigned to each entity to differentiate what the lines and points represent in a particular map. Meridian 2 and Strategi have different set of identifiers, for example in Meridian 2, code 3000 identifies motorway, but in Strategi, code 5310 is used instead.
There are some obvious drawbacks of the approach use by either of these maps. Firstly, although single carriageways and dual carriageways are distinguished (in Strategi, code 5323 and code 5320 is used respectively), they do not include information regarding direction of the traffic. Secondly, both maps present unnecessary geographical details excessively. So if their approaches are to be adapted, then the prototype might no longer be schematic.

2.3.2 SATURN

Saturn or Simulation and Assignment of Traffic to Urban Road Networks is a transport planning software developed by Institute for Transport Studies (ITS) of The University of Leeds. The following are a few basic functionalities of Saturn:

1. Simulate traffic to help with the analysis of road infrastructure.

2. Capable of handling any network regardless of its size.

3. Simulate accurate model of every road junction.

4. Capable of editing the network.

Saturn is mainly used to simulate the effect of traffic changes whose assessment requires detailed examination of activities around road junctions. For that reason, the structured map used by Saturn is designed very differently from any of Ordnance Survey products. Saturn’s structured map is designed to focus mostly on junctions.

Each junction contains detailed information on all traffic coming in and going out of it. Some of the traffic details that could be included in a junction are: traffic turns which must “give way” to other traffics, right turn on a major road that intersects with other major traffic flows, traffic turns which merge with other turns, traffic turns which has an exclusive lane (figure 2.9) and “weaving” traffic turns which represents traffic that enters from and exits to minor road (figure 2.10).
This approach in representing a network could be interpreted as a directed graph, where junctions could be considered as vertices, and motorways as directed edges. The graph is a connected graph because every junction could be reached from any of the other junctions. In other words, for every pair of vertices there is a path from one vertex to the other (this in fact is the definition of a connected graph). For those interested in the details of Saturn’s architecture could find more information in Hall et al (1992)[14].

The design of the structured map used by the prototype produced in this project will closely resemble the approach used by Saturn. This design will be discussed in details in chapter 3 Representing Data in Structured Map.
2.4 Methodology

Software development requires a systematic approach. Elements of a good systematic approach in software engineering can be summed up as following:

- Thorough analysis of the problems and existing solutions.
- Design of the system.
- Implementation of the system.
- Testing the functionality of the system.

However, there are various different methods as to when and how these individual elements should be carried out. Usually, a methodology is chosen based on which phase of a development process takes precedence over the other phases. For example, when building a safety critical software, Prototyping might not be the best method to use because all the above elements are crucial, and prototyping tends to neglect the design and implementation process.

This following chapters briefly discuss some of the existing methodologies and chooses an appropriate one to be carried out in the design of the program for this project.

2.4.1 Waterfall Model

The Waterfall Model is one of the oldest and most well-known software development methodology. It classifies a development process into different phases, as shown in figure 2.11.

In this methodology, the development process strictly progresses from top to bottom linearly. This is analogically-represented by the waterfall, as it is easier to go down but incredibly difficult to go up (See figure 2.11). Each phase is a prerequisite for the next one, therefore, each phase has to be accomplished in order to supply the input needed by the next phase.
Figure 2.11: The waterfall model of software development life cycle (Boehm, 1988)[15].

Obviously, the major drawback of this methodology is backtracking. Once the development is in a particular phase, it is very costly to go back to any of the previous phases, even though backtracking is what is required. As a result of backtracking, the development cost and time would increase disproportionately depending on how far back the development process would have to go.

One solution to this problem is by stopping any attempt to backtrack. Once a phase is completed and approved, the development process should not be allowed to alter that phase. However because of this, the end product might become obsolete as many of the fundamental requirements, which were discovered during the process of the development, were not implemented.

Obviously, like many other software development, backtracking is almost unavoidable in this project. So due to this significant drawback, Waterfall Model is not going to be employed in developing the Schematic Bus Route Map Editor.


2.4.2 Prototyping

This is a methodology where the developer would build a prototype which mimics the “real” program. Users would then experiment with the prototype and give feedback to the developer. This method would enable the developer to better understand the requirements as well as enable users to get an insight of the final program. Learning curve in using the program would also be lessened, as users actively involved in the development process and therefore could anticipate of the changes to come.

According to Avgerou (1998)[16], prototyping is perceived differently by different developers. Some of the definitions could be generally classified as the following:

- **Explanatory**, prototype is used to obtain knowledge of the requirements.

- **Experimental**, prototype is used to test newly added or modified feature of the system.

- **Incremental**, prototype is used as the basis which the program would be build upon, starting from few core functionalities {	extit{incrementing}} to a more and more complex system.

- **Evolutionary**, prototype is used as an approach to deal effectively with modified system environment.

To save cost and time, the developer would often neglect to consider crucial issues such as scalability, performance, security and efficiency of the prototype. Obviously these issues do heavily effect usability and functionality if the final system is build by re-using components from the prototype, which is frequently carried out in {	extit{prototyping}}.

For these reasons, prototyping might not be a favourable methodology to be used in building a safety-critical software.
2.4.3 Rapid Application Development

Rapid Application Development or RAD is a process modelling methodology that strongly emphasises on making use of parts from previous system. In the context of this project, RAD is obviously not suitable, as there is no previous system from where components could be salvaged. Therefore this methodology will not be discussed any further. More information regarding Rapid Application Development could be found in Pressman(2000)[17].

2.4.4 Incremental Model

The Incremental Model is one of the many methodologies that adapts the approach used by Evolutionary Model. Evolutionary Models recognise that not all phases of a development process could be conducted all at once. Many of these models are derived from Boehm’s Spiral Model (See figure 2.12), details could be found in Boehm(1988)[15].

![Figure 2.12: The Spiral model of software process][15].

The Incremental Model is similar to the Incremental Prototype proposed by Avgerou(1998)[16] (see
section 2.4.2). However, this model follows the conventional life cycles, where each development phase is completed before the next one is started. At each increment, a working version of the software is delivered. The deliverable starts with a few core functionalities, then it evolves into a more and more complex system with each increment.

Essentially, this methodology has many similarities with prototyping. However unlike prototyping, the Incremental Model delivers a working version of the software with each iteration.

Moreover, Pressman(2000)[17] stated that: “When you encounter an impossible deadline that cannot be changed, the incremental model is a good paradigm to consider”. Therefore considering the time frame of this project and workloads from other modules, the Incremental Model will be employed in building the Schematic Bus Route Map Editor.
Chapter 3

Representing Data in Structured Map

The Schematic Bus Route Map Editor is a program that helps user to view and design bus routes from scratch. It uses a structured map as input data. It also uses structured map to represent routes and bus stops created by the Editor. Therefore it is necessary to discuss how this data is represented in the structured map for both motorways and bus routes.

3.1 XML

Nowadays, Extensible Markup Language (XML) is widely used for representing digital information (More elementary information about XML could be found in first chapter of Goldfarb(2002)[18]).

Using XML to represent a data has a lot of advantages. Firstly, as the name suggest, XML is easily extensible. XML could be designed to describe any type of data, such as letters, molecular data, transport schedules, astronomy star constellations, etc.
Secondly, XML is standardised and nonproprietary.

Thirdly, XML is text-based. It describes data by encapsulating it in between a *start element* and an *end element*. For example: `<First Name>Darwin</First Name>`. Therefore though XML is machine readable, it is also much more human readable than most normal text-based data.

Fourthly, the Editor program produced by this project, like many other cartography software, would assume that the format of input from a structured map is accurate. However, unlike most text-based data, XML document could be automatically validated by checking if the structure of the document matches its Data Type Definition or *DTD*. Hence, a program could refuse to parse an XML document if it is found to be invalid.

Ultimately, XML is only useful if the data it represents could be processed. These processes (more commonly known as **parsing**) might involve reading the data from an XML document or outputting a set of data onto an XML document. In XML, parsing itself is a standardised process (which makes XML even more appealing) and the two most widely used API for parsing XML are Document Object Model or *DOM* and Simple API for XML or *SAX* (More information in Goldfarb(2002)[18]). Fortunately, almost all modern programming languages such as Java, Python and C++ have a package or library that implements parsing. This means a developer does not need to build his own parser every time he needs to process an XML document, therefore this is considered to be the most important advantage of using XML.

On the other hand, using XML is not without its drawbacks. A slight disadvantage is that normally an XML document is much larger in size than most non-XML text-based data. This is due to the fact that each record in XML has to be encapsulated by a start and an end element. Larger size document means that if XML is used in a distributed system, it could put additional strain on the network. This drawback however, does not concern the Editor program therefore it is not taken under consideration. Besides, all the advantages discussed above could more than compensate for this minor drawback.

Therefore the Schematic Bus Route Map Editor will utilise XML to defined its structured map for both roads and routes.
3.2 Structured Map for Roads

In order to allow the prototype to effectively switch between viewing options, roads and routes would have different formats of structured maps.

3.2.1 Road and Junction

It is obvious that any structured map that is utilised to manipulate bus routes has to contain information about roads and junctions. In this prototype, all junctions are simply represented as an intersection point. Differences between types of junctions, such as ring road, traffic light, roundabout, etc., will not be addressed. Similarly, all motorways will be modelled as a simple dual-carriage way with traffic flow going from and to both directions.

Some of the different approaches used in representing these data have been covered in section 2.3.1 and section 2.3.2. This section goes on to discuss how the Schematic Bus Route Map Editor will represent its structured map by adapting a combination of approaches used by Ordnance Survey products and SATURN.

As discussed earlier, the prototype is manipulating a schematic map. Therefore, every road is represented as a straight line with no curve displayed on the map. However, the prototype will deviate slightly from Harry Beck’s design by allowing roads to be in an angled other than 45° angle (diagonal). This adjustment is made to give the prototype a cutting edge map; a map that exists between a precise geographical map which might cloud the big picture by its overwhelming details, and a strictly schematic map which might be simplified too far from the real world. In other words, this adjustment will offer the prototype just the right amount of detail.

Since every road is a straight line, it could be modelled as a line between 2 points, a start point and an end point. Each point has to be either a junction or a coordinate of the motorway. This means a 4-way junction, as shown in figure 3.1, is denoted by 4 motorways which have one similar point. The coordinates of this point has to be identical.
The following example shows how the data of figure 3.1 would look like:

```
<map>
  <road name="A">
    <x1>-10</x1> <y1>-10</y1> <x2>389</x2> <y2>467</y2>
  </road>
  <road name="B">
    <x1>389</x1> <y1>467</y1> <x2>400</x2> <y2>370</y2>
  </road>
  <road name="C">
    <x1>389</x1> <y1>467</y1> <x2>542</x2> <y2>467</y2>
  </road>
  <road name="D">
    <x1>389</x1> <y1>467</y1> <x2>150</x2> <y2>850</y2>
  </road>
</map>
```

Note how each of the above “road” element has at least one point (either $x_1,y_1$ or $x_2,y_2$) that is identical with the other 3 “road” elements. In this particular example, the coordinate of the 4-way junction in figure 3.1 would be point (389,467). Also notice that all roads are contained by a “map” element and each road element has to have a “name” attribute, i.e. A, B, C and D. This “name” attribute corresponds to the street’s name and it is not unique. Therefore, if 2 parts of a road is separated by a junction, they could share the same name to indicate that both parts form the same road.
In term of graph theory, it is apparent that each pair of x and y coordinates in the structured map is a vertex. So since a road is a line between 2 adjacent vertices, then every road is an edge of the graph. It should also be noted that the graph is an undirected connected graph, as there are no vertices without an edge.

In conclusion, the following rules define how the structured map would represent motorways and junctions:

- Every structured map has exactly one “map” element. This element encloses all “road” elements.
- A motorway is denoted by at least one “road” element.
- Each “road” element must have a “name” attribute to stored the name of the motorway and 2 pairs of x,y coordinates, namely x1, y1, x2 and y2, to define its starting and ending coordinate.
- A junction is represented by a pair of x and y coordinate which are used as start or end coordinate by at least 3 different “road” elements.

### 3.2.2 Landmark

It is crucial for digital cartography software to be able to display landmarks. However, it could be argued that landmarks might obscure the overall view of a map and not suited to be implemented in a
schematic map. On the other hand, some landmarks are absolutely definitive to a map, for example: if the prototype is to display a map of Paris then the Eiffel Tower should be displayed in that map because the Eiffel Tower is the definitive landmark of Paris. For this reason, the prototype is implemented to display landmarks and as a result, the structured map needs a way to represent landmarks.

To illustrate how landmarks could be represented in the structured map used by this prototype, trees are going to be used. Every structured map contains exactly one “tree” element and a tree is defined by a pair of coordinates, namely x and y, enclosed within the “tree” element.

```xml
<tree>
  <x>35</x> <y>483</y>
  <x>44</x> <y>426</y>
</tree>
```

For example, the XML above would tell the prototype that there are 2 different tree entities, one is located on coordinate (35,483) and the other one is located on coordinate (44,426).

### 3.2.3 Area

In representing areas such as administrative area or boundaries, the structured map is going to adapt the approach used by Meridian 2 and Strategi (see section 2.3.1). A polygon is used to denote an area, permitted to be of any sizes. Such polygon is defined by a set of points as illustrated by figure 2.7 in section 2.3.1. But unlike Meridian 2 or Strategi, every polygon must have a name associated with it, which will be displayed by the prototype later on.

```xml
<region>
  <area name="LEEDS">
    <px>496</px> <py>462</py>
    <px>567</px> <py>436</py>
    <px>566</px> <py>392</py>
    <px>499</px> <py>372</py>
    <px>446</px> <py>387</py>
    <px>412</px> <py>411</py>
    <px>439</px> <py>440</py>
  </area>
</region>
```
The above example specifies 2 different entities of an “area”. One area is entitled as LEEDS which has 7 boundary points to make up for the polygon “LEEDS”. The other area is entitled BRADFORD and it only has 5 boundary points encapsulating the area.

To summarise, the following is how the structured map would define an area:

- Every structured map has exactly one “region” element. This element encloses all “area” elements.
- An area is denoted by exactly one “area” element.
- Each “area” element must have a “name” attribute to store the name of the area and at least one pair of x,y coordinates, namely px and py.

### 3.2.4 Data Type Definition

In order to aid the program in validating whether a given structured map is in the correct format, a Data Type Definition is required. The prototype will be designed in such a way that a DTD is compulsory for every “Road Structured Map” in order for it to be loaded into the prototype.

The following is the DTD which complies with all the entity’s definitions specified in previous sections:

```xml
<!DOCTYPE root [
  <!ELEMENT root ( map, tree, region )>
  <!ELEMENT map ( road* )>
]>
```
3.3 Structured Map for Route

3.3.1 Bus Route

A bus route is fundamentally a line entity which made up of several individual lines. As described earlier, a road is consisted of 2 points which denote its begin and end coordinates. Therefore, if these points are considered as vertices in graph theory and motorways as its edges, then every bus route must be a path of the graph.

The following is an example of how the structured map will represent this path:

```xml
<routes>
  <route name="Route Name 1" color="#101010">
    <x1>454</x1> <y1>-10</y1> <x2>480</x2> <y2>112</y2>
    <x1>498</x1> <y1>172</y1> <x2>480</x2> <y2>112</y2>
    <x1>498</x1> <y1>172</y1> <x2>449</x2> <y2>223</y2>
    <x1>499</x1> <y1>335</y1> <x2>449</x2> <y2>223</y2>
    <x1>499</x1> <y1>335</y1> <x2>499</x2> <y2>377</y2>
    <x1>566</x1> <y1>406</y1> <x2>499</x2> <y2>377</y2>
  </route>
  <route name="Route Name 2" color="#808080">
    <x1>454</x1> <y1>-10</y1> <x2>480</x2> <y2>112</y2>
    <x1>498</x1> <y1>172</y1> <x2>480</x2> <y2>112</y2>
    <x1>498</x1> <y1>172</y1> <x2>449</x2> <y2>223</y2>
    <x1>499</x1> <y1>335</y1> <x2>449</x2> <y2>223</y2>
    <x1>499</x1> <y1>335</y1> <x2>499</x2> <y2>377</y2>
    <x1>566</x1> <y1>406</y1> <x2>499</x2> <y2>377</y2>
  </route>
</routes>
```
Each “route” element has a “name” attribute and a “color” attribute. The “name” attribute represents the name of the bus route and the “color” attribute is an integer that indicates in what colour should the bus route be displayed by the program. Within the “route” element, each set of x1, y1, x2 and y2 element defines a line and a combination of all these lines will assemble the bus route. However, since each route is a path, all these lines have to be connected to one another. Therefore each line shares at least one similar x,y coordinate with another line.

To summarise, the following rules define how the structured map would represent bus routes:

- Every “Route Structured Map” has exactly one “routes” element. This element encloses all “route” elements.
- A bus route is denoted by exactly one “route” element.
- Each “route” element must have a “name” attribute to stored the name of the bus route and a “color” attribute to stored the colour of the bus routes in integer format. This integer format is derived from representation of 24 bits binary, where bits 0-7 are blue, bits 8-15 are green, bits 16-23 are red and bits 24-31 are alpha.
- Each “route” element must also have at least a set of x1, y1, x2 and y2 elements, to define the road passed by the route. Subsequent road has to be connected to the previous one, by matching either x1,y1 or x2,y2 with its own x1,y1 or x2,y2 coordinates.

3.3.2 Bus Stop

In a “Route Structured Map”, bus stop is defined similarly with how landmark is defined in a Road Structured Map (See section 3.2.2). Every bus stop is represented as a pair of x and y coordinates as shown below.
The data above specifies that there are 3 different bus stop entities, located on (496,398), (867,694) and (743,72).

3.3.3 Data Type Definition

The following is the DTD for “Route Structured Map” which complies with the entity’s definitions specified in previous sections:

```xml
<!DOCTYPE root [ 
<!ELEMENT root ( routes, busStops )>
<!ELEMENT routes ( route* )>
<!ELEMENT route ( (x1, y1, x2, y2)* )>
<!ATTLIST route name CDATA #REQUIRED>
<!ATTLIST route color CDATA #REQUIRED>
<!ELEMENT x1 ( #PCDATA )>
<!ELEMENT y1 ( #PCDATA )>
<!ELEMENT x2 ( #PCDATA )>
<!ELEMENT y2 ( #PCDATA )>
<!ELEMENT busStops ( (x, y)* )>
<!ELEMENT x ( #PCDATA )>
<!ELEMENT y ( #PCDATA )>
]>
```
Chapter 4

Prototype Implementation

4.1 Overall view of the System

A fully functional prototype of the Schematic Bus Route Map Editor is made up of several components. Figure 4.1 roughly illustrates the interaction between components within the Schematic Bus Route Map Editor. Some of the “not so explicit” components, such as the processing of structured map data, will be discussed in later sections (section 4.2).

As can be seen in figure 4.1, the prototype is initiated by specifying an input road structured map. Unfortunately, this project does not cover automated construction of such structured map. Therefore, if a company or an organisation wishes to use this application, work has to be done in advance to generate structured maps that suit its needs.
4.2 System Components

After designing the structured maps for both motorways and bus routes, the next logical step would be to implement them into the prototype.

4.2.1 Reading Data from a Structured Map

Similar to many digital cartography software, in order for the Schematic Bus Route Map Editor to design or edit a bus route, it needs to be initiated with a structured map as an input. The prototype would assume that the structured map is in the correct format. However since the structured map is specified in XML format, the prototype could perform basic XML validation by testing the structured map against the appropriate DTD (as explained in section 3.1).

Once the input structured map is validated, the prototype will start to parse all entities such as road names, road coordinates, tree coordinates, area names, etc. Parsing is performed using DOM, where all the data would then be stored into different arrays according to the type of data it is representing. As a result, when parsing is completed, all data would be grouped by its type, for example: all road names...
are stored in one array and all road coordinates in another. This data could then be relate back to one another by matching its array’s index number.

This approach is by no means the most efficient way to parse the structured map. Scalability would be a major issue if the structured map is too large. Especially if it is even larger than the size of the memory (although considering today standard of computer memory, this case might be uncommon). A proposed solution to this problem will be discussed in Evaluation chapter (chapter 5).

4.2.2 Prototype Display

As shown in figure 4.1, the application interface is designed to display the routes overlaying on top of the roads. Therefore if either one of the Roads Data or Routes Data is modified then the display will be updated to reflect these changes.
In addition to allowing user to switch between roads view and routes view, the prototype is also capable of zooming in and out of the map as well as moving around it. This functionality is commonly found in many Digital Cartography software.

Moving across the display or zooming in/out of the prototype could be performed simply by manipulating the arrays containing information regarding motorways and routes. A simple validating algorithm, that lets the prototype to verify whether each object is contained by or intersects the viewing plane before drawing it onto the canvas, would be sufficient. If an object is contained within the viewing plane then it obviously needs to be translated onto a correct position and scaled into a correct size.

For example, suppose figure 4.3 is how the prototype interface is split. If a user is zooming into region A then the prototype could decide which objects are to be displayed by calculating whether that particular object exists within the boundary of A.

Due to how the prototype is designed, switching between roads view and routes view is even simpler. If any or both of the views are switched off, then the prototype will simply omit the need to process the arrays containing the data.

4.2.3 Selection and Navigation Process

In this prototype, a route is essentially a combination of connected roads. So to create a route, users need to select one or more roads to indicate the selection of roads that make up for the bus route. To simplify the next part of explanation, this set of selected roads will be referred to as a “would-be” route.
Figure 4.4: Screen shot when both roads and routes are displayed.

Figure 4.5: Screen shot when only routes are displayed.
When creating a bus route interactively, users could only add a road to the would-be route if either one of its start/end point is identical to the start/end point of the currently selected roads. For instance, in figure 4.6 if road 1, 2, 3, 4, and 5 are already selected, the only road that could be added to this would-be route has to have a start/end point that matches point A or B. This process should be true regardless of the order in which the roads 1,2,3,4,5 were selected.

![Figure 4.6: A route of 1, 2, 3, 4 and 5 roads with A and B as begin and end points.](image)

This functionality could be implemented by providing a stack or an array that stores the start and end points of a would-be route. Everytime a user tries to add a road x to the would-be route, it is validated against the stack. If x is permitted to be added then the stack should be modified by replacing the matching point within the stack with the other point that makes up x. In effect, this will correctly update the stack with a new start and end point every time a road is added to a would-be route. Obviously, if the would-be route stack is empty, (ie. no road is currently selected) then x would just simply be added to the selection, and both start/end points of x will be store into the stack.

Similar operation could be performed to de-select a road. If a user selects a road y which had been previously selected, then the start/end points of y will be compared against the stack of start/end points of the would-be route. If no match is found, then the prototype will deny the user from de-selecting y. However, if a match is found then y would be de-selected and the stack would replace the matching point with the other point which makes up y. This, in effect, will update the stack correctly with a new start/end point.

After the selection process, if the user decides to create a route out of the selected roads then all the information regarding the route would be passed into arrays containing **Routes Data** (see figure 4.1). This, in effect, will force the prototype to update its display.
4.2.4 Exporting/Importing Routes Data

Both these methods are fairly straightforward. The capability to export is implemented by creating an XML document builder based on DOM. This builder would then pass all the necessary information concerning bus routes and bus stops, which are stored within Routes Data, into a well structured XML document. During this process, a correct DTD will also be passed into the structured map to ensure that the “Route Structured Map” could be validated by the prototype when it is imported later on.

In regards to importing, this could simply be done by modifying the same XML parser that parses the Roads Structured Map. So that instead of storing the information into Roads Data, it would store them into Routes Data. Therefore after the import process, the Routes Data would be modified. This, in effect, would trigger the prototype to update its display.

4.3 Prototype Incremental Process

Since Incremental Model is employed as the methodology, the prototype will start with few core functionalities then expand to a more and more advanced prototype.

As can be seen from the project schedule (section 1.3.2.1), the goal of a basic prototype is to be capable of handling a structured map properly. At this stage, the structured map is overly simplified such that it only contains data concerning road coordinates. An XML parser is created to aid the prototype to store all these information into an array. The basic prototype only allows the map to be looked at from a single universal view.

Then with the next few increments, the prototype is modified to accept a more complex structured map with road names and trees. The next step is to implement the functionality to detect mouse activities so that if a road is selected then it is going to be highlighted appropriately. The prototype could handle this event by capturing the mouse pointer position and compare it against all the road coordinates to determine whether the pointer position is within boundary of any of the road. This process is clearly not very efficient as the number of computation needed would increase linearly with the number of
the roads. A possible solution to this problem would be discussed further in chapter 5 *Evaluating the Prototype*.

The next prototype allows a road to be de-selected if it has been previously selected. Then the next stage is to validate if a road is connected to the route before appending it. The prototype could also create a bus route out of all selected roads. In this prototype, viewing options, which let user to switch between roads and routes as well as zooming and moving across the map, is implemented (details in section 4.2.2).

In the final prototype, the crucial capability to export/import routes data into/from an XML structured map is added. As a final touch, the functionality to export the “currently viewed” map into a standard image file is also implemented. This functionality gives users the flexibility to choose whether only roads or only routes or the combination of the two is exported by simply switching between view. The image format PNG or *Portable Network Graphics* is chosen over other formats, such as jpeg or gif, simply because it is lossless and non-proprietary.

### 4.4 Test Plan

Testing is a crucial phase of a development process. Since Incremental Model Methodology is employed, testing is carried out on each deliverable throughout this project. The following test plan uses the *Boundary Value Analysis* of the *Black Box* approach.
<table>
<thead>
<tr>
<th>No</th>
<th>Test</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Load a valid “Road Structured Map” (roads only)</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>2</td>
<td>Load a valid “Road Structured Map” (roads and trees only)</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>3</td>
<td>Load a valid “Road Structured Map” (roads, trees and regions)</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>4</td>
<td>Detect and refuse to load a “Road Structured Map” with invalid format</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>5</td>
<td>Detect and refuse to load a “Road Structured Map” which does not comply to its DTD</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>6</td>
<td>Select a road</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>7</td>
<td>Selecting a road twice, to de-select it</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>8</td>
<td>Select the road, which was de-selected previously</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>9</td>
<td>Validate road selection process, so that un-connected road could not be added to the currently selected roads</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>10</td>
<td>Create bus routes</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>11</td>
<td>Create bus stops</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>12</td>
<td>Zooming into the display</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>13</td>
<td>Zooming out of the display</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>14</td>
<td>Moving horizontally across the display</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>15</td>
<td>Moving vertically across the display</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>16</td>
<td>View roads only</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>17</td>
<td>View routes only</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>18</td>
<td>View both roads and routes</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
</tr>
<tr>
<td>19</td>
<td>Assign a name to a route</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>20</td>
<td>Assign a colour to a route</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>21</td>
<td>Input special character or number as a route name</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>No</td>
<td>Test</td>
<td>1&lt;sup&gt;st&lt;/sup&gt;</td>
<td>2&lt;sup&gt;nd&lt;/sup&gt;</td>
<td>3&lt;sup&gt;rd&lt;/sup&gt;</td>
<td>4&lt;sup&gt;th&lt;/sup&gt;</td>
</tr>
<tr>
<td>----</td>
<td>----------------------------------------------------------------------</td>
<td>----------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
</tr>
<tr>
<td>22</td>
<td>Input empty string as a route name</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>23</td>
<td>Returns error message if users refuse to assign a name when creating a route</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>24</td>
<td>Returns error message if users refuse to assign a colour when creating a route</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>25</td>
<td>Change view to “roads only” and export the display into a png image file</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>26</td>
<td>Change view to “routes only” and export the display into a png image file</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>27</td>
<td>Change view to “roads and routes” and export the display into a png image file</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>28</td>
<td>Export currently displayed routes into a “Route Structured Map”</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>29</td>
<td>Add DTD when exporting a “Route Structured Map”</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>30</td>
<td>Import the previously exported “Route Structured Map” into the application</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>31</td>
<td>Detect and refuse to import a “Route Structured Map” which does not comply to its DTD</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>32</td>
<td>Return error message when a “Road Structured Map” is expected to be loaded but a “Route Structured Map” is given</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>33</td>
<td>Return error message when a “Route Structured Map” is expected to be loaded but a “Road Structured Map” is given</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
</tr>
<tr>
<td>34</td>
<td>The descriptive caption (legend) is displayed correctly each time a route is created</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td>Pass</td>
</tr>
</tbody>
</table>
Chapter 5

Evaluating the Prototype

5.1 Evaluating Aims and Minimum Requirements

The aims and minimum requirements of this project were defined in section 1.3.1.1 and section 1.3.1.3 respectively. The stated minimum requirements are the basic criterions which required to be achieved, in order to determine whether this project had satisfied its aims. The following presents the assessment of the minimum requirements:

1. Present a survey on 2 forms of existing structured maps, examining how data are represented in each one of them. - Requirement met

The whole project started by conducting a research on Ordnance Survey structured maps. This research discovered that Meridian 2 and Strategi are the two most suitable structured maps to be adapted, since they concentrate more on road and traffic representation. To get a better idea, another investigation into how Saturn represents its structured map was also conducted. The result
of both investigations have made up for how the structured map of this project was designed. Section 3 examined how data are represented in both structured maps.

2. Design and model how data would be represented in an artificial schematic structured map. This map would later be processed by the application. - **Requirement exceeded**

The design decisions made on the structured map is primarily based on the research above. For instance, “road” and “route” entities were modelled by adapting Saturn’s structured map, and “area” entity was modelled by adapting Ordnance Survey structured maps.

The project also analysed and considered using XML format to depict its structured map. Both the pro and cons of XML are fully explored during the process.

3. The program should correctly display the map stored within any given structured map, provided that the structured map is in the right format. - **Requirement exceeded**

Given a valid structured map, the final prototype could display it properly without mis-interpreting the data. The final prototype also exceeded the minimum requirement by validating the structured map before it is loaded. Validation is carried out by comparing the structured map against its DTD. In addition, the prototype could also distinguish between “Road” and “Routes” structured maps by their formats, so that the prototype is incapable of mis-interpreting “Road Structured Map” as “Routes” and vice versa.

4. The program should allow users to interact directly with the map to create bus routes. - **Requirement exceeded**

The application not only allows users to create bus routes interactively, it also permits name and colour to be assigned to every route. This functionality is crucial, since a nameless and non colour coded route in a schematic map is unfavourable. In addition, bus stops could also be created interactively.

5. Conduct tests on the prototype to evaluate its functional behaviours, and consider possible and desirable extensions to the system. - **Requirement met**

Testing was carried out thoroughly with each deliverable of the prototype to ensure that its behaviours are as expected (see section 4.4). The scalability of the prototype and proposed extensions
to the project as a whole will be discussed in the later sections.

To summarise, the following is a brief outline of the project expansions and additional functionalities incorporated within the prototype:

- Present a detail analysis on XML, examining why it should be used as the format for the structured map.
- User could assign a name and a colour to a route when it is built interactively.
- The program allows user to create bus stops interactively.
- The program validates input structured map against its DTD and verifies its format to distinguish between “Road” and “Route” Structured Maps.
- The program is capable of exporting the currently displayed routes into a structured map file.
- The program should be able to import and display correctly any “Route” structure map that was previously exported.
- The program would allow thee user to alter the view of the map by switching on and off either the bus routes or the road map.
- The program provides alternative output ie. exporting the current display into a standard image file.

5.2 Evaluating Scalability of the prototype

Many of the functionalities and design decisions made during the implementation process are not the most efficient method. The following are some proposed modifications that could be made to reduce scalability problem of the prototype.
5.2.1 Parsing the Structured Map

In the real world, a structured map representing Geographical Information System data tends to be very large in size. Furthermore, the prototype of this project uses structured map in XML format, and XML is normally larger than other text-based data file. So if a given structured map is too large, the prototype might not be able to handle it properly. Therefore this section will propose a solution that might overcome this scalability problem.

There are 2 main API or Application Programming Interface for processing XML documents, namely Document Object Model (DOM) and Simple Api for XML (SAX).

DOM is an API standardised by World Wide Web Consortium (W3C). It returns a tree structured object based on the structure of the XML document. This tree structure is called a Document Object Model tree. DOM stores this hierarchical tree structure in memory, as a result, programs could gain access to the XML data in an instant. DOM also permits programs to modify the XML documents by offering capabilities to add and remove nodes within the DOM tree.

The prototype employs DOM in its parsing mechanism. Since DOM stores the whole XML data in memory all at once, this approach may not scale very well.

An alternative solution to this is to utilise SAX instead of DOM. SAX is developed by members of XML-DEV discussion groups in 1998. SAX is an event-based parser. It does not create a tree structure instead it parses the data as it finds them. This, in effect, eliminates the memory overhead and increases system performance.

However, using SAX is not without its downsides. Firstly, SAX could not give the “big picture” of the document’s structure. Secondly, SAX does not support data modification, in other words, SAX is unable to add or remove any data within XML documents. But if this approach is used to parse “Road Structured Maps”, this drawback does not effect the prototype at all since a “Road Structured Map” does not need to be modified. Nevertheless, it is important to note that SAX is not suitable to be used to export “Route Structured Maps” because of this drawback.
5.2.2 Road Selection Process

To identify which road is selected or de-selected by users, the prototype compares the position of the mouse pointer against all currently displayed roads. In worst case scenario, the number of comparisons needed is equal to the total number of roads. Therefore the computational time increases linearly with the growth of the number of roads. In big-Oh notation this could be expressed as \( O(n) \).

This obviously is not good enough for the prototype, as the fundamental functionality of the prototype is to create bus routes from scratch by selecting and de-selecting roads. A better technique to deal with this problem is by dividing the currently displayed area into regions (as shown in figure 5.1).

![Figure 5.1: Example of how the prototype screen might be split.](image)

When the road selection process begins, mouse pointer position is compared against each of these regions to determine which region the position of the pointer is. Assume that it is in region \( x \), then the position is again compared with all the roads that contained by or intersected with region \( x \). This technique eliminates the need to perform comparisons with all existing roads. Therefore computational time only increases logarithmically with the growth of the number of roads, ie. \( O(\log n) \).

It is apparent that the size of the regions need to be chosen carefully. If the regions are too small or too big, the technique might not be as efficient. For instance, let \( z \) be the position returned by the mouse. If the regions are too small then more comparisons needed to be performed on \( z \) to determine in which region \( z \) is located. In contrast, if the regions are too big, it is faster to determine in which region \( z \) is located but the number of roads in that region would also be large. Therefore \( z \) needed to be compared with all those roads to find out which road is selected by \( z \).
Moreover, a particular region might have more roads than others, an example of this is the city center area. A good way to maintain efficiency is by dividing a region into even smaller regions depending on how clustered it is. Figure 5.2 shows an example of region E of figure 5.1 being divided into 4 smaller regions.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>F</td>
</tr>
<tr>
<td>G</td>
<td>H</td>
<td>I</td>
</tr>
</tbody>
</table>

Figure 5.2: Example of how a region could be split further.

### 5.3 Proposed Extensions

There are many capabilities that should be incorporated with the prototype to better fulfil the objectives of this project. The following will discuss a few proposed extensions to the prototype as well as the project:

1. The prototype should **associate each bus stop with at least one bus route**. A preferable method in illustrating this association is by drawing the bus stops precisely on top of the routes it is associated with. This method complies with the Harry Beck’s schematic map design. But, due to how the structured map is modelled, this association might not be achievable. Hence fundamental changes need to be made to the “Route Structured Map” design in order to implement this extension.

2. Currently if there exists 2 or more bus routes which happen to share the same road, the prototype will draw them superimposed on one another. The order of the drawing is determined by which route was created first. This approach causes the routes to be hard to differentiate, besides it also deviates from Beck’s design. A proposed solution to this problem is by **drawing the routes in parallel to minimise overlapping**.
3. Implement the capability to delete and modify bus routes.

4. Conduct a research on GML or Geography Markup Language, developed by the Open Geospatial Consortium (OGC).

5. This project could be expanded to include a functionality to aid users with the semi-automated construction of “Road Structured Map”. One possible technique is by recording the position of 2 consecutive mouse clicks. Users could then decide whether these points are the start/end points or the end/start points of the road, or both (in other words, a 2-way road). Afterwards, these position could be exported into a structured map and used as the basis for creating bus routes. Precision might be a problem with this approach because users might not be able to pinpoint the exact location of a road using a mouse. However, since this is a Schematic map, geographically correct position is not a concern.

6. As mentioned earlier, one of the assumption made in design was that all roads are dual-carriageway with traffic going in both direction. Obviously this assumption is unrealistic. However, due to the design of the structured map, this extension could be implemented without major difficulty.

   As explained in section 3.2.1, a “road” entity is denoted by a start and an end point. So if the “start point” is considered as the starting point of a 2-dimensional vector for example point A of figure 5.3, and the “end point” is point B then the one-way road is the vector $\vec{AB}$.

   ![Figure 5.3: An AB vector.](image)

   On the same note, a 2-way road could be expressed by 2 “road” entities with opposing direction, ie. vector $\vec{AB}$ and vector $\vec{BA}$.

7. If the above expansion is implemented, then it is possible to design a bus route which “turns back” on the same road. This expansion could be realised without changing any fundamental design of the “Route Structured Map”.

48
8. Implement a **polynomial time function to discover the shortest route between 2 locations** on the map without users intervention in defining the route. Taking into consideration on how roads are modelled, this function could be regarded as the *Single Source Shortest Path Problem*. A proposed solution to this problem is the infamous *Dijkstra’s Algorithm*[19].
Bibliography


Appendix A

Personal Reflection

During my placement year, I have spent most of my time in integrating and minimising administrative tasks by building database applications to be used across the Intranet. In my quest to explore other subjects of Computer Science, I have decided to leave the familiar territory of scripting, databases and HTML and try to carry out a project which is closely related to Geographical Information System. Whether this decision was a good idea, is still remain to be seen.

GIS is a completely new field of study for me. Therefore in my pursuit to understand this project better, I have seek the help of many people, from Deborah Allon and Anna Clough to Dr Bill Lythgoe and Dr David Milne of The Institute for Transport Studies. Their advice and suggestions have helped me significantly, especially in understanding the concept and applications of Structured Maps.

This project has made me appreciate the design and complexity of Structured Maps. I have also recognised how a design could vary from others depending on its applications. This, in effect, gave me a general understanding of subjects revolving around Digital Cartography.
At the beginning of the project, I had major difficulty in finding any existing Structured Maps which, in my opinion, suited to be adapted. As a result, the progress of the whole project almost came to a halt. However, in one of the comments made by the assessor on my Mid-Project Report, is that I should have made simplifying assumptions on some design elements in order to make progress. This is indeed very accurate. Based on this advice, I managed to advance and develop a format of “Structured Map”. As the project proceed, some of the unrealistic assumptions were eliminated along the way.

Therefore, I believe and would like to advise other fellow students who are developing an application; if some features or design elements are bringing the whole project to a standstill, then it might be a good idea to make assumptions to simplify them in order to make progress. These assumptions could then be eradicated one at a time, once the project is moving forward.

During the implementation process, I believe that I have chosen a suitable development methodology for this project. The Incremental Model insists on producing a working deliverable with each increment. This, in effect, means that the prototype could be tested carefully and thoroughly with each increment. Since I was also under pressure from other modules, I frequently forced to temporarily abandon this project and then got back to it later on. This has always been a problem for me, because I need a substantial amount of time to go back to a piece of work once it is neglected. Fortunately, given that each deliverable of the development process is a working prototype, it helps me to go back to the project much quicker by analysing and testing the deliverable.

At the end of this project, I have discovered many information on how this project could be better carried out. For instance, when designing a text-based data structure, XML is a good format to consider. It seems obvious, but somehow I did not thought of that until it was pointed out to me by the assessor. If only I have investigated on XML this earlier, then numerous attempts in designing a “flat file” structured map could have been omitted.