Table of Contents

[1.0 Introduction - 2 -](#_Toc480819705)

[1.1.0 Problem drivers - 2 -](#_Toc480819706)

[1.1.1 Problem to be solved - 2 -](#_Toc480819707)

[1.1.2 Project Purpose - 2 -](#_Toc480819708)

[1.1.3 Project Objectives - 3 -](#_Toc480819709)

[1.1.4 Limiting the Project Scope - 3 -](#_Toc480819710)

[1.1.5 Outline of plan - 3 -](#_Toc480819711)

[1.2.0 Output Summary - 4 -](#_Toc480819712)

[1.2.1 Requirement Specification Document - 4 -](#_Toc480819713)

[1.2.2 Design Specification Document - 4 -](#_Toc480819714)

[1.2.3 Visualisation platform - 4 -](#_Toc480819715)

[1.2.4 Testing Documents - 5 -](#_Toc480819716)

[2.0 Literature Review - 6 -](#_Toc480819717)

[2.1 Initial Development - 6 -](#_Toc480819718)

[2.2 Java Data Structures - 6 -](#_Toc480819719)

[2.3 JavaScript - 6 -](#_Toc480819720)

[2.4 D3.js - 7 -](#_Toc480819721)

[3.1.0 Methodology - 7 -](#_Toc480819722)

[3.2.0 Code - 7 -](#_Toc480819723)

# Introduction

## 1.1.0 Problem drivers

Organizations using Business Intelligence, use raw data to extract significant information to make decisions, spot business problems and boost their performance. The problem arises when these information are inaccurate and they lack quality, leading them to decrease efficiency in business and production. For such cases, a model should be carefully designed using validation techniques to ensure that the output will meet the appropriate criteria for each case study. Firstly, developers ensure that the conceptual model is valid by gathering information from the stakeholders and address the main problem. Then they build a computerised model to implement a solution for the problem. There might be cases when the model’s output behaviour is accurate but there will be times when the model is tested with different input it will produce inaccurate results. This causes confusion and requires the developer’s team to test their code again and again until they spot the main problem.

## 1.1.1 Problem to be solved

Knowledge engineering group (KEG) is a team of researchers, employed by the University of Brighton to work on large projects worldwide. The group consists a team of developers who were assigned a project to develop business intelligence platform for the Southeastern Railway. This project requires the use of raw data such as train timetables, stops and destinations. KEG’s analyst realised the rail company had many issues with their historic data as the trains had many delays or technical problems with their train fleet. A part of this project was to create a route finder that will be hosted to the rail’s company website. The developers using Java, they have already build a model that computerised the routes using the data provided by the rail company. Unfortunately, their code was not efficient and it produced inaccurate outputs. What irritated them the most was the time they needed to spend debugging the code to identify the problem. Their need was to find a different and most efficient way to understand and identify their problem as the deadline for their project was pressing them.

## 1.1.2 Project Purpose

The purpose for this project was to create a simulation model that illustrates clearly the whole implementation of their code and visualise the process how data are manipulated and extracted to their final form. By breaking down the code into different modules and visualising them in a different environment will give a clear understanding to the developers how their code executes using different data sets. The whole model should be re-programmed using JavaScript and D3.js. The code should be translated carefully to serve the main functionality as the original one. The main target for this project is to not identify and create a better version of the previous model but to develop a new environment that will visualise the process and to be tested using the many data sets provided by the company. This will give a clear insight to the group’s developers to identify potential mistakes caused on the operational process or even more lack of data quality. The new model will use these data for the input and it will produce a new output that it will be compared and tested with the correct results provided by the KEG’s staff.

## 1.1.3 Project Objectives

Objectives targeted as follows:

* Create an interactive and fully functional visualisation environment that receives a csv file and produces a new data set with the correct outputs
* Fully understand and translate Java code and data structures to JavaScript
* Understand the requirements and protocols provided by the KEG’s developers
* Visualise each component of code
* Test every module separately
* Test the system thoroughly and diminish any bugs

• Code the system in such a way so near future changes can be done effortlessly

## 1.1.4 Limiting the Project Scope

The scope of this project was to develop a visualisation environment to add data and illustrate the process until the expected output is extracted. In order to decide upon the functionality, the author researched extensively the available visualisation frameworks to find the best course of action. Using the data the rail company provided along with the research done and the frequent meetings with the developer’s team, enable the author to construct a plan using an agile software engineering model. Moreover, by examining the Java code, the author was able to decide and choose more technical stuff like the data structures required to enable the same functionality of the original code, combined with the visualisation components.

## 1.1.5 Outline of plan

**Research:** The first course of action was to collect as much information as possible from the KEG developers to identify their main issue and understand their code. Then the author researched available visualisation frameworks and decided the most suitable for this project.

**Design:** After collecting all the necessary information and deciding upon an agile model, a plan was created to design the basic model of visualisation which included the flow chart of the code transformed into modules. Moreover, functional requirements were created for the system and discussed with the director of Knowledge engineering group until an agreement was reached. JavaFx was used for the creation of the visualisation environment but was replaced with D3.js later when the team decided to host the platform on a web browser.

**Implementation**: The implementation started as soon as the team decided to use JavaScript libraries. The first stages of the implementation were to convert the Java code to JavaScript and use queries to store the output to MySQL. Later, all csv files were converted to JSON format to ease the process and avoid relation database systems.

**Testing:** The testing phase consists of two different parts. Firstly, each node on the visualisation platform must be checked it produces the same result as the original code. After making sure the operational process was functioning properly, train data were used to produce the routes for each train. Then it was compared with the KEG’s output to ensure both workflows were producing the same results. After all checks and tests are completed, KEG will be able to use the simulation model to identify mistakes and improve their code.

## 1.2.0 Output Summary

In this section of the project, the author will be talking about the major documents produced during the project.

## 1.2.1 Requirement Specification Document

Recipient: Knowledge Engineering group, author

File type : Microsoft Word Document

The first document that was created was the requirement specification document which was crucial in the development of the system since it was created to record and understand the client needs. Moreover, it can be used as a reference point for what the system needs to achieve.

This document contains the requirements specification document with a list of all the functional and non-functional requirements for the system approved by the director of KEG. These were developed after an extensive information gathering and analysis which included methods like interviewing the group members, using a questionnaire to identify their basic needs and understand the process taken to develop the original code for the route finder part of the system.

All the documents created can be found in **APPENDIX A**

## 1.2.2 Design Specification Document

Recipient: Knowledge Engineering group, author

File type : Microsoft Word Document

The second document created was to help even further with the development of the project and again act as a reference for what the system should look like.

This document contains designs of the visualisation as well as flow-charts diagrams that were used to structure the functionality of the code. These were derived from the information gathered by the author and the Java code of the route finder.

all the documents, drafts, diagrams created can be found in **APPENDIX B**.

## 1.2.3 Visualisation platform

Recipient: Knowledge Engineering group

File type : .rar file

This is the finished product that will be used from the group as their visualisation environment to check their code efficiency and execution process in each step, filtered based on each row on the data set and each route created. This folder contains a repository with the raw data, a repository with the original code and the new code written in JavaScript and a repository containing all the files needed for the visualisation including the data in JSON format.

Some of the outputs create are shown on **APPENDIX C**

## 1.2.4 Testing Documents

# 2.0 Literature Review

This section will detail what books, articles and resources the author had to read in order to be able to undertake this project. This section has been broken up to into four main categories which include Initial Development, Java data structures, JavaScript and d3.js.

## 2.1 Initial Development

Project planning and control was one of the modules the author attended during his university studies, therefore after researching development models online and reading books the author decided to work on Agile model. To do so the user reviewed all the module materials to find more information about development models and their advantages. After reading the book “Software Engineering” (Sommerville, 2010) the author found three development models that all suited his needs and were straightforward to understand. The Waterfall model which was eventually rejected, is the more standard development model used were the developers need to split the project into separate phases considering the fundamental process activities of specification, development, validation and evolution (Sommerville, 2010). An alternative option would have been the increment model which is based on the idea of developing an initial implementation, exposing this user comment and evolving it through several versions until an adequate system has been developed (Sommervile, 2010). This was then considered both by the author and the client due to the fact that constant communication was feasible as both were living in Brighton and they could plan regular meetings. At the end both agreed to use Agile model as the requirements of this project might change over the time. In contradiction with the waterfall model there was no need to have a strict plan as the development started early and by the regular meeting there was a feedback through the whole process. The quality of the software was improved using the extreme programming methodology by delivering a small version of the final product each week or month to the client and discussing changes or improvements.

## 2.2 Java Data Structures

During the research and design process the author spent a lot of time interviewing KEG’s developers to understand their code. The data structures that were used for the Java code were Lists and ListIterator. Again, the author had previous experience with data structures form his 2nd year module called Data Structures and Algorithms. To refresh his memory and make the most sensible decision the author spend time reading the Oracle’s documentation about the ListIterator interface and its methods. All the data in the Java code were manipulated using the methods from that interface, such as hasNext(), hasPrevious(), next() and previous(). After understanding these methods, the author tested the code by printing the results on the console to understand clearly their functionality.

## 2.3 JavaScript

The decision to choose d3.js to implement the visualisation for this project led the author to choose JavaScript as the programming language for translating the Java code. D3 is a library written in JavaScript so it will be much easier to combine the logic of the code with the visualisation environment. Because Java is different from JavaScript, there was a risk that the functionality of the system could not meet the requirements. To ensure this problem would not exists, the author researched ways to implement a class in JavaScript having the same functionality as the ListIterator interface. By reading posts in StackOverflow, there were many developers that have recreated this interface using plain JavaScript. After the long discussion the author had with the KEG’s director, they decided to not use this method, instead the author should create his own way to manipulate data in the list. By reading articles on StackOverflow on how to manipulate data inside lists or arrays the author concluded to use a simple for loop and create variables that will decide which objects in the list will be considered as current, previous and next.

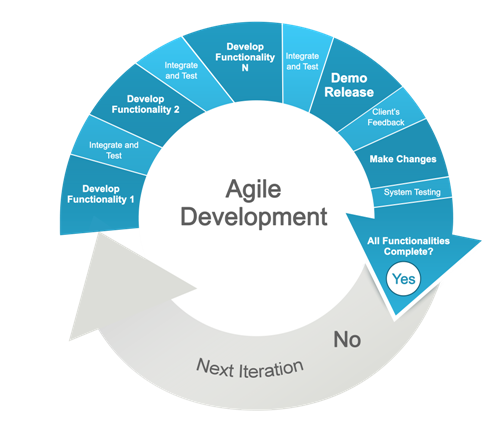
## 2.4 D3.js

D3 is a small library for great visualisations and its widely discussed as one of the best techniques to visualise almost anything. Like programming, d3 needs a lot of practise to understand its methods and get familiarised with it. The author during his research process started spending time studying Mike’s Bostock website which is the author of d3 - Available at : <https://bost.ocks.org/mike/> (Accessed: 27/10/2016). After understanding the fundamentals of this library, the author researched the official website of d3 Available at: <https://d3js.org> (Accessed: 10/11/2016) which provided an amazing documentation for all types of visualisations and their code. By using the code provided by these websites and d3’s GitHub profile, the author experimented with many types of visualisations and decided to use the template provided by Mike Bostock on collapsible trees. By using this tree, the code could be easily illustrated by breaking it down into modules and visualise them as tree nodes.

# 3.1.0 Methodology

This chapter will describe the methods used and work done by the author over the course of the project to achieve its purpose.

## 3.1.1 Software Development Model

The software development model that was used by the author is the Agile model. It is a model that satisfies the stakeholders through early and continuous delivery of valuable software. It welcomes changing requirements, even late in development. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. (Mike Beedle, 2002). This model was ideal for this project as the main problem of the stakeholders was to visualise their code and the end-goal was not clearly defined. The requirements change as the members of KEG could decide a new or modify existing features on the visualisation environment in any time during the process. The great communication between the author and the team is ideal for this model as the stakeholders will provide feedback during the end of each iteration.

# 3.2.0 Code