Parallelisation of Data Visualisation

School of Computer Science
BSc Computer Science with Business & Management with Industrial Experience

Author: Karolis Skripka
Supervisor: Dr Richard Neville
May 2015
Contents
List of figures ........................................................................................................................................... 3
List of tables ............................................................................................................................................ 4
Abstract .................................................................................................................................................. 5
Acknowledgements ............................................................................................................................... 6
1 Introduction .......................................................................................................................................... 7
1.1 Chapter overview ............................................................................................................................ 7
1.2 Problem overview ........................................................................................................................... 7
1.3 Proposed solution ............................................................................................................................ 7
1.4 Alternative solutions ....................................................................................................................... 7
1.5 Report structure ............................................................................................................................... 7
1.6 Chapter summary ............................................................................................................................. 8
2 Background .......................................................................................................................................... 9
2.1 Chapter overview ........................................................................................................................... 9
2.2 Data science overview ..................................................................................................................... 9
2.3 Visualisation in data science .......................................................................................................... 10
2.4 Chapter summary ............................................................................................................................ 11
3 Design ............................................................................................................................................... 12
3.1 Chapter overview .......................................................................................................................... 12
3.2 Software engineering ...................................................................................................................... 12
3.2.1 Waterfall .................................................................................................................................. 12
3.2.2 Agile ........................................................................................................................................ 12
3.2.3 Short iterations ......................................................................................................................... 12
3.3 Requirements gathering ................................................................................................................ 13
3.3.1 Features list .............................................................................................................................. 13
3.3.2 Hierarchical task analysis ........................................................................................................ 14
3.3.3 System class overview ............................................................................................................. 14
3.3.4 Hierarchical task analysis and system class relationship .......................................................... 15
3.4 Chapter summary ............................................................................................................................ 15
4 Implementation .................................................................................................................................... 16
4.1 Chapter overview ............................................................................................................................ 16
4.2 Implementation overview .............................................................................................................. 16
4.3 Integration environment ................................................................................................................ 16
4.4 Selection of programming language ................................................................. 16
4.5 Development environment ............................................................................... 17
4.6 Prototype table .................................................................................................. 17
4.7 Parallelisation ................................................................................................. 17
4.8 Real time processing algorithm ..................................................................... 18
4.9 Data flow diagram .......................................................................................... 18
4.10 User interface ................................................................................................. 19
4.11 Application in use .......................................................................................... 19
4.11 Chapter summary ......................................................................................... 21

5 Evaluation and Testing ...................................................................................... 23
5.1 Chapter overview ............................................................................................. 23
5.2 Testing overview .............................................................................................. 23
5.3 Unit testing ........................................................................................................ 23
5.4 Integration testing ............................................................................................ 24
5.5 Acceptance testing .......................................................................................... 24
5.6 Tests during short iterations ........................................................................... 24
5.7 Chapter summary ............................................................................................. 24

6 Conclusions ......................................................................................................... 25
6.1 Chapter overview ............................................................................................. 25
6.2 Development of ideas ...................................................................................... 25
6.3 Project evaluation ............................................................................................ 25
6.4 Skills attained .................................................................................................. 25
6.5 Future improvements ....................................................................................... 26
6.6 Chapter summary ............................................................................................. 26

7 References ........................................................................................................... 27
List of figures

Figure 1. An illustration of typical raw data collected during research.............................................. 9
Figure 2. An illustration of typical data analysis .................................................................................. 10
Figure 3. An illustration of converging and volatile variables ................................................................. 10
Figure 4. Project’s hierarchical task diagram ......................................................................................... 14
Figure 5. Artefact’s system class diagram .............................................................................................. 15
Figure 6. Pseudo algorithm for parallel real time charting................................................................. 18
Figure 7. Data flow diagram during real time processing algorithm .................................................. 19
Figure 8. Artefact’s Graphical User Interface ....................................................................................... 20
Figure 9. Screen displaying detected input parameters ......................................................................... 20
Figure 10. Artefact in real time operation mode with 14 parallel inputs .............................................. 21
Figure 11. Single visualised variable under investigation ..................................................................... 21
Figure 12. Code coverage of visualising parts ....................................................................................... 23
List of tables

Table 1. Gantt chart of undertaken short iteration schedule ......................................................... 13
Table 2. Project’s feature list ........................................................................................................ 14
Table 3. Project’s prototype table ................................................................................................ 17
Abstract

An investigation has been made into the data analysis and data science fields. The lack of tools capable of analysing exponentially growing data has been identified. An artefact has been developed to solve the problem and to enable scientists to review data in real time. One approach to analyse data is to build a pipeline which processes data sequentially. Another approach is to analyse data in parallel. The project has explored both approaches and it was chosen to utilise the parallel data analysis. During the artefact’s development, multiple parallelisation techniques such as threads and tasks have been explored. The produced artefact is capable of visualising static and dynamic data in real time (as data is produced). The artefact for data analysis utilises the most efficient data analysis technique – data visualisation. The artefact has been developed using the Agile short iteration approach. To ensure the artefact’s testability, interfaces and factory methods have been utilised in the code. Possible further improvements such as GPU processing have also been analysed.
Acknowledgements

The author would like to offer extensive thanks to his supervisor, Dr. Richard Neville, for weekly meetings, mentoring and support during the research and development stages. A big part of this project could not have happened without the involvement of agile course teaching staff, who laid the foundation for the author’s understanding of the software’s development. The author would also like to thank the researchers who contributed their valuable time and allowed themselves to be interviewed.
1 Introduction

1.1 Chapter overview
This chapter provides an overview of the existing problem (§1.2). §1.3 & §1.4 discuss alternatives and a proposed solution to the problem. Chapter provides an overview (§1.5) of the report’s structure.

1.2 Problem overview
During the research process, data is gathered, analysed and interpreted to spot trends, cycles and patterns [1]. Accurate data analysis is crucial to ensure correctness of the research results. It is estimated that global data size will grow by 4300% by 2020 [2]. An increased amount of complex data will require a solution capable of efficiently analysing data. Artificial intelligence algorithms are utilised to reduce the amount of data requiring analysis [3]. Currently, essential data sets are analysed by manually creating visualisations. With exponential growth in data size manual analysis will become infeasible to perform in a timely manner. This problem requires an automated solution which is proposed by the undertaken project and discussed in §1.3.

1.3 Proposed solution
The project proposes to create an automated solution for data analysis, through the identification of two types of data analysis performed by the researchers. The first type of data analysis is performed after data has been gathered, while the proposed solution addresses this by providing the ability to visualise static data. The second type of data analysis is performed at the same time as data is gathered, with this type of data growing in an exponential manner. The artefact utilises parallel processing to support this type of dynamic data. Without utilisation of parallelisation it would be infeasible to visualise multiple inputs at the same time that the data is produced by the experiment. The project solution divides input data into multiple streams to enable the researcher to choose relevant data and discard irrelevant data during the analysis. This allows for data filtering and saves computational resources. The alternatives currently available are discussed in §1.4.

1.4 Alternative solutions
There exist multiple solutions for static data analysis but there is lack of tools for automated data analysis. Tools like IBM InfoSphere Streams (Streams) [4] are capable of providing automated data analysis. Streams and similar tools are more complex and provide more features than allowed by the project’s proposed solution. However, these tools are inaccessible to the project’s targeted audience (solo and small group researchers) due to specific hardware requirements such as dedicated servers for analysis software.

The following chapter §1.5 will present the structure of the report.

1.5 Report structure
To demonstrate the development of the process report that has been undertaken, the following breakdown is provided.

- Introduction. This chapter provides overview of the problem and proposes solution to the problem.
• Background. This chapter provides an overview of the project’s domain, the domain’s importance and evolution of the domain.
• Design. This chapter describes the software engineering approach undertaken and displays implemented design stages.
• Implementation. This chapter focuses on evolution of the artefact by displaying involved algorithms and system design.
• Evaluation. This chapter evaluates the artefact’s code testability and evaluates acceptance by project stakeholders.
• Summary. This chapter concludes the undertaken project process and provides suggestions for possible further project improvements.

1.6 Chapter summary
This chapter has explained the existing problem and proposed a solution. It gave a brief overview of the report’s structure. Alternatives to the proposed solution have been evaluated and advantages of the project’s artefact presented. The following §2 will provide an in-depth overview of the problem’s background.
2 Background

2.1 Chapter overview
This chapter provides background on the data analysis. It explains how computer scientists utilise data science to identify trends and patterns. Examples of data visualisation are provided and different types of data interpretation explained.

2.1 Data analysis overview
Data analysis can be defined as a process of applying mathematical and logical techniques to describe data [5]. As far as science exists as a field, data is used to draw hypotheses and analyse the performed experiments [6]. Key parties interested in data analysis are scientists, librarians, statisticians and computer scientists [7]. This project is concerned with computer scientists using data analysis for drawing conclusions for performed experiments [5]. One of the fields directly aligned to data analysis is data science; this is discussed in §2.2.

2.2 Data science overview
Data science was first mentioned by William S. Cleveland in 2001 [8]. It combines data analysis together with the field of computer science. Research data is analysed by converting raw measurements (Figure 1) into more readable artefacts such as charts (Figure 2). Charts allow for the display of relationships between dependent and independent variables. Dependencies are utilised to detect trends, patterns and to recognise anomalies. The utilisation of visualisations in data science is discussed in §2.3.

| 0.263038 | 0.728443 | 0.9559 | 0.061081 | 0.058445 | 0.086171 | 0.288283 |
| 0.463916 | 0.984632 | 0.648215 | 0.842938 | 0.337117 | 0.967343 | 0.176884 |
| 0.080042 | 0.823119 | 0.392028 | 0.695062 | 0.646908 | 0.797494 | 0.526581 |
| 0.446566 | 0.177387 | 0.190687 | 0.942954 | 0.339182 | 0.173431 | 0.074049 |
| 0.306968 | 0.548237 | 0.405835 | 0.163365 | 0.776851 | 0.637883 | 0.79356 |
| 0.436873 | 0.316564 | 0.144384 | 0.771582 | 0.74713 | 0.042144 | 0.586919 |
| 0.844076 | 0.340526 | 0.342915 | 0.489884 | 0.048205 | 0.865955 | 0.076358 |
| 0.096147 | 0.223911 | 0.723796 | 0.559906 | 0.415044 | 0.014768 | 0.440616 |
| 0.527908 | 0.358185 | 0.546188 | 0.947121 | 0.229745 | 0.580831 | 0.828709 |
| 0.107605 | 0.483563 | 0.237247 | 0.48669 | 0.627036 | 0.901704 | 0.017063 |
| 0.124978 | 0.776328 | 0.082421 | 0.804781 | 0.568295 | 0.459839 | 0.576129 |

Figure 1. An illustration of typical raw data collected during research. Each column represents a different output variable. Each row shows output during next measurement time (with the oldest data at the top, newest at the bottom).
2.3 Visualisation in data science

During experiments, quantitative data is produced. Data is required to be summarised and benchmarked to ensure its validity [11]. Visualisation techniques are used to detect two types of behaviour. The first type of detection is concerned with variables converging to maximum or minimum. A converged variable identifies the success or failure points in the experiment. Understanding these points allows for repeating the experiment under conditions which bring an expected outcome [11]. The second type of behaviour detection is concerned with spotting variable volatility. Volatility can be used to identify stability of data and to predict ranges over which data changes unexpectedly. This type of pattern is utilised in future prediction algorithms [12]. An example of variable converging and an example of variables being volatile can be seen in Figure 3. In the provided example, trends can be identified using logical reasoning. Without the ability to visualise data it would be difficult to identify trends from raw measurements using only mathematical calculations. The provided example displays one variable analysis. Most performed experiments consist of multiple data points which require similar analysis.
2.4 Chapter summary

This chapter analysed different types of data analysis. It explained how raw measurements are converted into more interpretable artefacts. The chapter also provided background in data analysis and the data science fields. It included an explanation of how visualisation is utilised to interpret the data. §3 discusses software engineering and the design techniques undertaken in the delivery of this project.
3 Design

3.1 Chapter overview
This chapter gives an overview of the project design. The overview is broken down into two parts: i) software engineering and; ii) system design. The software engineering section (§3.2) contrasts two software engineering approaches and gives the reason as to why a specific approach (§3.2.2) has been chosen; whereas, the system design section (§3.3) explains the requirement elicitation process and shows how to elicit requirements for the high and low level system design of the artefact.

3.2 Software engineering
Software engineering is an essential part in the software development process because it enables the design to be planned; it also lays down the methodology required for the software evolution [13]. Choosing the right software development methodology can be a distinguishing factor between the project succeeding or failing due to the project team not being able to adapt to the required changes [14]. Changes can occur for multiple reasons, like changes in the operating environment, new discoveries and regulations or wrongly gathered requirements in the beginning. Without the correct techniques in place it could be too expensive to perform the required changes [22]. The next section describes one of the considered software engineering techniques for the undertaken project.

3.2.1 Waterfall
Following Waterfall’s software development, the ability to move from one stage to the next without repeating any stage twice is expected. The Waterfall method is normally used when a project has a strict timeline and clearly defined requirements. The project under consideration had a strict timeline, however, it did not possess any clearly defined requirements, rather the requirements were able to evolve during the development stage. Waterfall methodology’s assumption that requirements are fixed and incapable of dealing with change [15] lead to further analysis of available software engineering approaches. The most commonly chosen alternative to the Waterfall method is discussed in §3.2.2.

3.2.2 Agile
Agile guidelines and recommendations are defined by the Agile manifesto [16]. Agile follows an iterative development approach. Every iteration could be considered as a Waterfall project on its own. Agile practice recognises and puts emphasis on requirements changing during the development process. The project undertaken required this type of practice to ensure timely delivery and the ability to satisfy the project requirements. The practice itself is composed of multiple instruments like extreme programming (XP), scrum, and short iterations [17]. Different instruments or combination of instruments were chosen per project basis. Due to short fallings in the Waterfall development methodology and the flexibility of the Agile development methodology, Agile methodology was chosen for the project under consideration. The project utilised Agile’s practice of short iterations which is discussed in §3.2.3.

3.2.3 Short iterations
Short iterations practice breaks a project timeline into fixed same length cycles. Each cycle consists of requirement gathering, implementation and testing stages. Having the right size of iterations provides the benefit of gaining timely feedback from stakeholders and ensuring that the software meets the latest requirements. As suggested by the course lecturer on The University of Manchester COMP33711 Agile course, a two week long iteration length for the undertaken project should be
chosen. At the end of each cycle, a new project prototype is released and presented to one of the stakeholders – the assigned project supervisor. The project’s iteration plan together with corresponding prototypes is shown in Table 1. The project’s requirements and the analysis of the requirements are discussed in §3.3.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Prototype 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototype 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototype 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototype 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototype 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototype 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototype 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototype 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prototype 9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Gantt chart of undertaken short iteration schedule. Holiday periods have been excluded from the table.

### 3.3 Requirements gathering

As suggested by the project’s chosen Agile methodology, requirements gathering were performed at the start of the project. During the requirement gathering process, interviews were held with project stakeholders. Project stakeholders being the project supervisor and postgraduate students performing research. As proposed by [18] the aims of the interviews were to discover requirements and potential threats to the project delivery. During each iteration, requirements for that iteration are clarified to ensure the persistence of the requirement’s relevancy. Initially gathered project requirements are discussed in §3.3.1.

#### 3.3.1 Features list

The feature list is a high level requirements specification. The list is expressed as a table with concise statements [19]. During the undertaken project, the features list has been expanded with an additional column of priority. The latter serves as an indicator for the order in which research should be undertaken. The project’s feature list is provided in Table 2. Based on the feature list, user tasks are generated. To ensure correct behaviour of the tasks, analysis needs to be performed; this is described in §3.3.2.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>#F1</td>
<td>Means of inputting data to be displayed</td>
<td>1</td>
</tr>
<tr>
<td>#F2</td>
<td>Means of defining input data axis</td>
<td>3</td>
</tr>
<tr>
<td>#F3</td>
<td>Means of labelling data in visualisation</td>
<td>5</td>
</tr>
<tr>
<td>#F4</td>
<td>Ability to visualise data in parallel</td>
<td>2</td>
</tr>
<tr>
<td>#F5</td>
<td>Means of displaying data in 2D</td>
<td>6</td>
</tr>
<tr>
<td>#F6</td>
<td>Means of inputting real time data</td>
<td>4</td>
</tr>
<tr>
<td>#F7</td>
<td>Means of discarding invalid data</td>
<td>7</td>
</tr>
<tr>
<td>#F8</td>
<td>Means of discarding stale data during real time visualisation</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 2. Project’s feature list.

### 3.3.2 Hierarchical task analysis

Hierarchical task analysis (HTA) is a way of breaking down systems into separate tasks based on a user’s interactions with the system. An analysis represents a high level overview of the system from the user’s perspective. HTA helps developers to understand the flow of user actions needed in order to achieve the system’s goal [20]. With the help of a hierarchical task diagram, developers are able to partition the application code into decoupled parts. Decoupled parts can be enhanced or replaced without impacting the application flow or the user experience. Figure 4 displays a project artefact hierarchical task diagram for the main function “analyse data”. To ensure the project’s decoupled infrastructure system, a class diagram based HTA was designed; this is discussed in §3.3.3 and §3.3.4.

![Figure 4. The project’s hierarchical task diagram. The figure depicts actions necessary to perform analysis of data.](image)

### 3.3.3 System class overview

A system class diagram is a Unified Modelling Language (UML) type diagram which describes system classes and the interaction between them [21]. The diagram is a static representation of the system which is created before the code implementation stage. In the early development stage, the diagram allows work to be performed in separated parts and describes dependencies between classes, allowing prioritisation of development for the artefact’s parts. During the software enhancing cycle, a system diagram is used to identify classes requiring extension or change. The next chapter displays how the HTA diagram is linked to create a system class diagram.
3.3.4 Hierarchical task analysis and system class relationship

Figure 5 displays a project artefact system class diagram. To display relationships better, the diagram has been simplified by excluding classes which implement interfaces and classes which are responsible for the artefact’s configuration. The diagram utilises standard C# notation of separating interfaces with <<>> and interface classes with the name beginning with a capital I. The system diagram was created before the code implementation stage. To ensure code maintainability and an ability to change based on the business domain - the diagram is based on an HTA diagram. This ensures that applications can be changed based on the operating domain, without impacting multiple classes. Links between the HTA and system class diagram can be identified as follows. Different operation modes (online and offline) are directly mapped to RealTimeCharter and StaticFileCharter classes. The HTA task of reading data is represented by the IUtilities class, which is responsible for displaying popups and retrieving a user’s input regarding axis requiring visualisation. The visualisation aspect itself has been abstracted into the ILineChart interface. This abstraction allows for future support of other type visualisations and provides support for different types of data storage. In respect it allowed RealTime and StaticFile charter classes to store data in different data structures while utilizing the same classes and code for displaying the data. To enable different types of visualization to be performed per input basis the factory method was utilized for initializing classes responsible for display visualisations.

![Diagram](image)

Figure 5. Artefact’s system class diagram. The figure displays the relationship and interactions between classes. The interface implementing classes and configuration classes has been emitted from the figure to ensure a simplified view.

3.4 Chapter summary

This chapter provided the benefits and drawbacks of both the Waterfall and Agile software engineering approaches. It explained why, due to the involved stakeholders and specificity of the undertaken project, Agile and short iterations practices were chosen. The chapter presented the project’s high level requirements, which were mapped to the hierarchical task analysis diagram. The system class diagram has been displayed and the relationship to the HTA diagram explained. The design implementation is presented in §4.
4 Implementation

4.1 Chapter overview
This chapter provides the background on the artefact’s implementation environment and technologies. Alternatives are presented for mostly used data analysis tools into which the artefact could be integrated. The chapter presents the development environment and libraries involved in parallelisation. Different programming languages considered for the project are presented and the choice explained. The chapter’s implementation part covers all project prototypes and provides algorithm together with a data flow diagram for the artefact’s most complex operational mode.

4.2 Implementation overview
Implementation is the software development stage where the artefact’s actual code is created. During this stage, the system is built based on previously created designs (§3) and chosen development methodologies (§4.4 & §4.5). Most implementation stages fail due to changes in requirements [23]. To ensure success of the undertaken project it was developed using an iterative approach consisting of nine implementation cycles. Multiple cycles in the project ensure that the built artefact always satisfies the latest requirements. The following §4.3 chapter discusses the artefact’s operating environment and possible alternatives.

4.3 Integration environment
Based on the project’s requirement for specifically visualising experiments’ produced data it was chosen to investigate the most common experimentation analysis platforms. Experiments can be performed in two ways – manually or automated. During manual experimentation and research, individual records are kept in plain text form for further analysis. During automated experimentation and research, data is produced by algorithm or automated observation mechanism and is saved in plain text format (.csv) for further analysis [11]. These observations were made based on interviews with professors and postgraduate students at The University of Manchester. The most widely used platform for data analysis is Microsoft Excel. Microsoft Excel provides data sorting and plotting abilities which are essential for successful data analysis. A currently available alternative to Microsoft Excel is the application, Tableau, owned by Tableau Software Inc. [www.tableau.com]. Tableau is an enterprise aimed tool for data analysis, supporting data visualisation on a semi real-time basis, with native functionality for data shaping and merging. The drawbacks of Tableau are that it requires an expensive enterprise licence and the application is currently inaccessible outside of the enterprise field. To increase the likelihood of the project artefact’s wider adoption and familiarity for the end user it was chosen to integrate the artefact into Microsoft Excel. The different implementation languages supported by Microsoft Excel are covered in §4.4.

4.4 Selection of programming language
Microsoft Excel supports integrations in two languages – Visual Basic and C#. Visual Basic was first released for Windows in 1991 and in later years has been adopted for Microsoft Suite integration. Visual Basic language provides the fastest integration time into the Microsoft Excel platform, however, it has performance and versioning issues. The main factors against choosing Visual Basic for this project was lack of support for object oriented programming, lack of support in creating multi-threaded applications and Microsoft’s decision, back in 2008, to discontinue support for the language.
C# is a programming language which solves all of the Visual Basic issues. The language supports an object oriented programming approach, has extensive libraries and supports different ways of creating multi-threaded applications. These factors were key influencers in choosing C#
programming language for the undertaken project. To ensure fast prototyping, the latest development environment was chosen, which is explained in §4.5.

### 4.5 Development environment

The development environment used for this project was Microsoft Visual Studio 2015. It was chosen in order to leverage a Microsoft .NET v.4.5 framework. Using this framework allowed utilisation of multiple native Windows functionalities such as browsing a file system or operating with files. Utilisation of these functionalities allowed us to allocate more time to implementing complex features. Parallelisation in the project has been achieved by utilising libraries available in the framework. To ensure the optimised number of imported libraries and loops, it was decided to utilise the 3rd party plugin Reshaper for removing redundant code. The project artefact was developed and extensively tested in Microsoft Excel 2007. Due to using the latest technologies utilised in the artefact it was possible to integrate later Microsoft Excel versions without any additional code changes. Incremental steps for the artefact’s development are discussed in §4.6.

### 4.6 Prototype table

The project consisted of nine two week long iterations. Requirements changed three times during the development stage. Changes in requirements did not, however, impact features described in §3. The implemented prototype table with corresponding features is displayed in Table 3. To display progress in the artefact’s parallelisation the prototype table was extended with a corresponding number of threads involved in running the prototype. The implementation of different types of parallelisation is discussed in §4.7.

<table>
<thead>
<tr>
<th>#</th>
<th>Milestone</th>
<th>Description</th>
<th>FL Number</th>
<th>Tasks / Threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>P01</td>
<td>Runs in Excel environment</td>
<td>Artefact is able to start in Excel environment</td>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td>P02</td>
<td>Display simple data</td>
<td>Artefact runs with hardcoded data</td>
<td>#F5</td>
<td>1</td>
</tr>
<tr>
<td>P03</td>
<td>Statically loaded data</td>
<td>Artefact is able to load data from closed file</td>
<td>#F1</td>
<td>1</td>
</tr>
<tr>
<td>P04</td>
<td>Real time data display</td>
<td>Single data feed display</td>
<td>#F6</td>
<td>2</td>
</tr>
<tr>
<td>P05</td>
<td>Parallel visualisation</td>
<td>Parallel data inputs are displayed</td>
<td>#F4</td>
<td>5</td>
</tr>
<tr>
<td>P06</td>
<td>Input selection</td>
<td>Ability to select required axis</td>
<td>#F3</td>
<td>20</td>
</tr>
<tr>
<td>P07</td>
<td>Error handling</td>
<td>Ability to ignore misshaped / incorrect data</td>
<td>#F7</td>
<td>20</td>
</tr>
<tr>
<td>P08</td>
<td>Data shaping</td>
<td>Ability to discard data after set period</td>
<td>#F8</td>
<td>50</td>
</tr>
<tr>
<td>P09</td>
<td>Graphical User Interface</td>
<td>User can define configuration</td>
<td>#F2</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 3. Project’s prototype table. # - Indicates prototype number. Feature list number has been abbreviated to - FL number. Task/Threads column displays number of CPU threads involved in running the prototype.

### 4.7 Parallelisation

Parallelisation is a process where multiple computations are performed simultaneously [24]. It allows for the utilisation of multi core processors and also for the utilisation of GPU for parallel
computations [25]. Without parallelisation the project’s artefact would not be able to visualise multiple inputs simultaneously. Simultaneous visualisation of the data is essential to be able to perform accurate data analysis in real time. The project’s artefact underwent multiple different levels of parallelisation. The first few prototypes were screen blocking, single threaded applications. The fourth prototype introduced threads and thread pools to separate data input and data visualisation into separate processes. Investigation into threads led to the conclusion that writing artefacts with the possibility of performing unlimited parallel computations is not possible using threads. Direct threads’ implementation in C# is a low level approach method which provides manual control of threads’ execution and scheduling time. Scheduling is important to ensure efficient CPU usage, however, it becomes nearly impossible to perform scheduling manually with the rising number of threads. It was decided to utilise the Task Parallel Library (TPL) to solve these issues. TPL provides an abstraction level for writing parallel applications. Library is responsible for monitoring the activity of each task and performing scheduling based on the activity level of that task. This feature enables the project’s artefact to perform a bigger number of parallel computations than it would be possible using manual thread implementation. The algorithm utilised in parallel visualisation are discussed in §4.8.

4.8 Real time processing algorithm
The artefact supports two modes of operation – static and live. During static mode the artefact deals with static data – that is, previously ran and saved experiment results. Data is obtained from a closed static file. During live operation mode, the artefact deals with data growing in size. Data is produced by a 3rd party algorithm or application. The artefact is capable of integrating with any 3rd party application and does not require any changes in the integrated application. Read input data is broken down into multiple data storage containers to allow retrieval to be performed in parallel at different data reading rates. Utilising parallelisation requires fewer resources and creates an ability to manage resource allocation via Windows Task Manager; this ensures maximum resources are allocated for running the research application. The algorithm for performing parallelisation is displayed in Figure 6. Data flow of the algorithm is explained in §4.9.

```csharp
Function ChartRealTimeData(source)
{
    Create new task to constantly read data from the source
    Prompt user with detected axes to select inputs requiring visualisation
    For each user selected input
    {
        Create new data storage
        Create new chart
        Create new parallel task and link with data source and chart
        Start created task
    }
}
```

Figure 6. Pseudo algorithm for parallel real time charting.

4.9 Data flow diagram
Data flow during a real time algorithm’s performance is represented in Figure 7. The diagram explains how a data source is split and how data is stored in multiple thread safe containers. Split data is consumed by multiple parallel tasks. Each parallel task consists of two parts – data consumer
and the element responsible for data visualisation. Composing tasks of two units helps to decrease the size of the task, which ensures that the task is allocated equal processing time during the operation and no screen blocking or delays occur during the artefact’s execution. The interface of the artefact is displayed in §4.10.

![Data flow diagram during the real time processing algorithm. The block at far left represents data produced by a 3rd party application. The data produced is a solid unit containing multiple outputs. The algorithm splits the solid unit into multiple containers for multi-threaded access.](image)

4.10 User interface

The artefact’s user interface is displayed in Figure 8. The artefact merges into Microsoft Excel in the right hand side panel. The interface can be resized, closed and opened from Microsoft Excel’s top ribbon panel. Four units compose the artefact’s user interface. The top interface part is responsible for defining the artefact’s operational mode – static or real time data. The visualisation offset part is only active during live operational mode. Changing the offset parameter defines how many data points per input are considered to be relevant before data should be discarded. Suggested default value is 50 data points. Data points’ upper bound limit is based on the size of available operating memory (RAM) and the number of inputs needed to be visualised. Suggested default offset was selected after holding interviews with postgraduate students involved in data analysis. The third user interface part is responsible for identifying the data source file. The interface bottom part is responsible for launching and suspending the artefact operation. §4.11 provides an example of the use of the application.

4.11 Application in use

Standard usage of the artefact is as follows. User using Figure 8 interface selects operational mode and data input file. After pressing the start button, the user will be presented with Figure 9 interface displaying all detected inputs in the data source. After confirming variables which are relative to experimentation the artefact will start the visualisation process. An application visualising 14 live inputs is displayed in Figure 10. Depending on selected inputs counted, the artefact will automatically resize and arrange windows to ensure all inputs fit into available screen size. Each window has the capability of being closed, resized, and moved independently. Each individual chart can be zoomed into and specific point coordinates can be extracted using a mouse pointer as displayed in Figure 11.
Figure 8. Artefact's Graphical User Interface.

Figure 9. Screen displaying detected input parameters. The screen is used to confirm which variables are relevant and should be visualised by the artefact.
4.11 Chapter summary

The chapter presented the Graphical User Interface and gave an overview of standard usage for the application. To display evolution of prototype, the prototype table has been provided. A real time data processing algorithm has been explained using a pseudo algorithm and data flow diagram.
Alternatives in programming languages and integration environments have been explained and reasoning provided for the chosen choices. The chapter explained the importance of visualisation regarding the artefact and provided the history of parallelisation for the prototype. The following §5 will discuss the evaluation and testability of the artefact.
5 Evaluation and Testing

5.1 Chapter overview
This chapter presents the different testing methodologies (§5.3 to §5.5) available during the software development process. It also presents the differences and rationale of black box and white box testing. This chapter documents frameworks (§5.2) involved during the artefact’s testing stage.

5.2 Testing overview
During software development the artefact’s validity and correctness are evaluated by performing tests on the artefacts [26]. Tests execute part or the entire artefact under known inputs and conditions. The outputs are examined in order to identify errors in the artefact [27]. Testing can be broken into two types: i) white box, and ii) black box testing. During white box testing the developer utilises underlying implementation knowledge to enable test cases to be developed; these cover all possible execution paths in the testable unit and ensure that the unit under test will perform as expected with all possible inputs. During black box testing, the developer is not aware of any underlying implementation and focuses instead on testing elements based on the element’s purpose [28]. The benefit of black box testing is that it creates less overlapping tests, however, it might not cover all important test cases. For extensive software testing it is recommended that both white box and black box testing are undertaken. The white box testing that was undertaken is described in §5.3

5.3 Unit testing
Unit testing under normal conditions is white box testing methodology. Tests are performed on isolated separate parts of the system [29]. Most units under test are complex and dependent on other units. To ensure correct testing and isolation during project development, Moq\(^1\) framework was utilised. During unit testing, dependencies are replaced with stubs or mocks to enable a unit’s isolation. Substituted elements are able to track the number of times it has been called together with callable arguments. The history of calls being made provide the ability to test if underlying implementation of the unit behaves correctly. Testing underlying behaviour is essential for white box testing. NUnit v2 test syntax and test runner was utilised to enable tests being run in parallel. The artefact’s essential parts were tested with code coverage of over 80% (Figure 12). This shows that tests cover over 80% of available functions. A more abstract way of testing is integration testing which is described in §5.4.

![Figure 12. Code coverage of visualising parts. Column one represents the artefact under test. The two following columns display percentage of executed functions and lines of code during test execution.](image)

\(^1\)Moq framework allows for the creation of stubs and mocks of the real objects [30].
5.4 Integration testing
Integration testing tests how separate parts integrate with each other [31]. Integration tests can be performed either as white or black box testing. They ensure that an artefact’s parts operate in an expected manner and follow agreed contracts. During the project’s lifetime two sets of integration tests were performed. The first set of tests focused on testing classes and their dependencies on each other. This ensured that the parts integrated correctly and followed agreed standards. The second set of tests tested that the artefact integrated correctly with the operational environment – Microsoft Excel. The second set of tests ensured that the artefact was operational in Microsoft Excel without interfering with other parts of the application. The black box testing performed is described in §5.5.

5.5 Acceptance testing
Acceptance tests are a black box testing method. During acceptance testing, project stakeholders evaluate an artefact to ensure that it provides expected features [32]. The project’s nine releases have been evaluated by the future users of the artefact – lecturers and postgraduate students. Acceptance tests provide an important form of feedback which was taken into consideration during the following releases. The standard practice of testing in Agile methodology is discussed in §5.6.

5.6 Tests during short iterations
The project has been implemented using an Agile short iteration practice. At the end of every iteration, a new prototype has been released, together with new, corresponding set of tests. This type of approach enabled creation of a set of regression tests which can be executed during the release to ensure no previous functionality has been broken. Due to tests being developed at the same time as the artefact, instead of after the artefact has been delivered, high code coverage was able to be maintained.

5.7 Chapter summary
The chapter presented black box and white box testing methodologies. It provided background on different testing methods performed during an artefact’s development. The chapter discussed different programming libraries which have been utilised to ensure testing practices are performed following suggested guidelines. The next chapter (§6) will conclude the report.
6 Conclusions

6.1 Chapter overview
This chapter summarises how the artefact has been developed based on the project’s requirements and specification. It describes the skills that the author attained during research and development stages. Possible future improvement areas are also suggested.

6.2 Development of ideas
The first prototype was developed in order to enable us to start visualising the data. This was extended into the next prototypes to enable us to view multiple data sources; this was with a view of enabling the researcher to gain a more detailed view of the data. Significant progress was made in the final prototype where parallel data processing was enabled in order to perform visualisation in real time. It was noted that parallelisation offered a time advantage over the previous prototypes. In an attempt to develop an effective visualisation tool, a fully parallel pipeline that input multiple data and visualised this data concurrently, was designed and implemented. In the following section §6.3 a full project evaluation is discussed.

6.3 Project evaluation
The project has been assessed against the project’s requirements and identified features described in §3. The project has achieved all of the high priority requirements and fulfilled all of the specifications. Additional suggestions such as GPU processing have been explored and it has been identified that it would be possible to utilise GPU for computations. Due to existing time constraints it was decided to leave this and similar improvements for other releases of the artefact. The success of the project’s deliverability would be attributable to the chosen software development practice (§3.2.2). Utilising Agile and short iterations allowed timely feedback loops which ensured that errors in the artefact could be identified and solved in the early stages. The artefact’s evolution during nine releases and utilisation of the design techniques ensure that final deliverability is ensured by following best programming practices. Utilised encapsulation and factory patterns allowed sustainable testing to be performed on the artefact (§5). Following §6.4 analyses, the author attained a number of skills during the project’s development.

6.4 Skills attained
The project contributed extensively to the author’s research and analysis skills. It provided an understanding of data analysis and the data science fields. Performing interviews with researchers provided the essential inner knowledge required to produce the artefact, which would be utilised after its release. Learned research techniques such as an HTA diagram or research relevance table served as the basis for efficient learning regarding the libraries and technologies required for the artefact development stage. The author’s technical knowledge improved extensively. Apart from learning a new development language (C#) and new development environment, the author also had an opportunity to explore GPU computations, which are the latest trend in the software industry. Exploration of the GPU computation will thus contribute to the author’s future employability. The requirements proposed by different prototypes led the author to learn three different ways of creating applications which utilise multi core processing. Prioritising the project’s testability led the author to learn new testing techniques involving mocking and substituting objects. These techniques allow for the isolation of objects under test, which is essential for performing correct unit and integration testing. Exploration and artefact integration into Microsoft Excel also provided
distinguishing knowledge between the author and his peers. Future artefact improvements are proposed in §6.5.

6.5 Future improvements
During the artefact’s development, a couple of possible future improvements have been identified. The first improvement could be utilisation of the GPU processing. This type of improvement would allow a bigger number of computations to be performed at the same time, which is not possible by utilising only CPU computations. The second type of improvement could be an addition of different charting types like histogram or point diagrams which would enable the analysis of more complex data. During the addition of charts it would be recommended that the ability to visualise data in 3D is implemented. Visualisation in 3D would allow for the exploring of dependencies for multiple variables in one plot. Depending on the artefact’s success it would be recommended that the market share of Linux and Apple IOs is explored between the researchers. If a sufficient need could be identified, then it would be recommended that the artefact adopt a different operating system.

6.6 Chapter summary
This chapter provided possible future improvements of the artefact. It analysed the author’s attained technical and soft skills. The project has been evaluated using the set features and requirements which have been achieved during the project span. The chapter ends the report and is followed by the reference list.
7 References


