Durham University

School of Engineering and Computing Sciences

Requirements Document

2010-11

Project Name: Durham Software Wind Tunnels

Team Number: 1

Team Members: Chris Salt, Amir Aliyu, Joe Rivett, Craig Condron, Andy Dykes

Document Information

Project Name:	Durham Software Wind Tunnels		
Prepared By:	Andrew Dykes	**Preparation Date:**	
Email / Phone:	andrew.dykes@durham.ac.uk		
Document Version No:	8	**Document Version Date:**	02/12/10

Version History

Ver. No.	Ver. Date	Revised By	Description

Table of Contents

1 Introductory Summary

1.1 Purpose

Our brief is to develop a piece of software to act as an intermediary between a user wishing to control or edit data from any one of the wind tunnels based in the engineering department of Durham university, and the command line programs which (depending on the nature of the individual program) either control the wind tunnel or preform some data manipulation action on a file containing pre-obtained data. The software must have a graphical user interface which is intuitive and error minimising.

The wind tunnels are controlled by a set of programs which have been written by the engineering department, these programs are currently run by calling them and passing parameters to and from the command line. The software must be able to run these programs, including creating any control files needed to run them. Along with this the software must also allow the user to add a new program to the list of runnable programs. This document relates to the whole of first edition of the software.

1.2 Document Overview

This document is concerned with the user requirements for the software. It is broken down in to multiple sections detailed below:

Analysis Process - This section considers what methods the team used to analyse the problem and problem space. This section includes interview methods, documentation analysis as well as more open problem analysis.

Analysis - This section details what information was obtained through the processes listed in 2.1. It gives a more in depth look at the problem and what solutions are possible.

Project Plan - This section details what deadlines (both external ‘hard’ deadlines relating to the customer, and internal ‘soft’ deadlines set by the team) are in place for the project. It includes a Gantt chart showing which tasks are to be carried out when. This section also details the roles allocated to the team and what they mean.

Hardware and Software Platforms - This section details what hardware and software is needed to run the program. It also lists what hardware and software would be needed to design a solution.

Functional Requirements - This section details the specific behaviors of the system and the features it must have. i.e. Functional requirements state what the system must practically do.

Non-Functional Requirements - This section details how the system must behave. i.e. Non-functional requirements state the characteristics of the system and how it must be.

1.3 Intended Audience and Reading Suggestions

This document is intended for: the project manager Nicola Bolland-Hill; the team involved in testing the software; the development team and Dr David Sims-Williams the customer who has requested the software. The testing and development team should focus on section 7. All others should read the document in order.

2.1 Analysis Process

To analyse the problem the we will be using multiple analysis techniques. These techniques are detailed below.

2.1.1 Interviews

Principally we will rely on the information to be obtained by interviewing several stakeholders related to the system. The team will speak to multiple users of the system, each of whom will ideally have differing amounts of experience related to the current system. Of the stakeholders to be interviewed, we will rely principally on the information provided by the systems main programmer/creator while giving adequate weight to what other users had say.

2.1.2 Documentation

Furthermore, we will have the Durham Software Wind-Tunnel (DSW) project mandate in order to decide on the constituents of the problem domain. The project mandate provides a background to the problem, providing information regarding the current system and its functionality.

In addition, we will make use of the DSW user manual. Reading through the manual, we should gain further insight into how the system works and greater clarity with regards to the context in which it is used. Moreover, the manual should provide further information with regards to the programs commonly used, in theory allowing us to compare and corroborate this with the information we will obtain from interviews.

With this analysis process we will devise the needs of the system. Using this information we can decide on suitable features that our implementation of the system must have. We will do this through further in depth study of our problem domain.

After having decided on a solution, we will devise suitable functional and non-functional requirements. The team will use the analysis from the problem domain, along with the information from our features coupled with several mind mapping solutions in order to come up with the system requirements.

2.1.3 Critical to Quality

To consider the project scope of the problem we will use CTQ (Critical to Quality) analysis. This method breaks down a fundamental project vision into a tree of smaller and smaller actions which must come together to realise the project vision. This method should significantly help to reduce any scope creep as the requirements will be mapped to the overall goal eliminating the need for additions.

2.1.4 Kano Modelling

To evaluate priorities for our requirements we will use the ‘kano model’. This method, developed by Noriaki Kano shifts the focus of a project from the users needs to the software functions, and specifically, the users’ response to functionality.

Putting the model in to practice involves a user filling in a questionnaire marking which response they would have to each requirement. There are 5 possible responses:

Attractive - This is a requirement which when met will deliver satisfaction, whilst when it is not met there is no dissatisfaction. For example if a yogurt pot has an easy peeling lid (not needed but will give satisfaction). These requirements will have moderate priority.

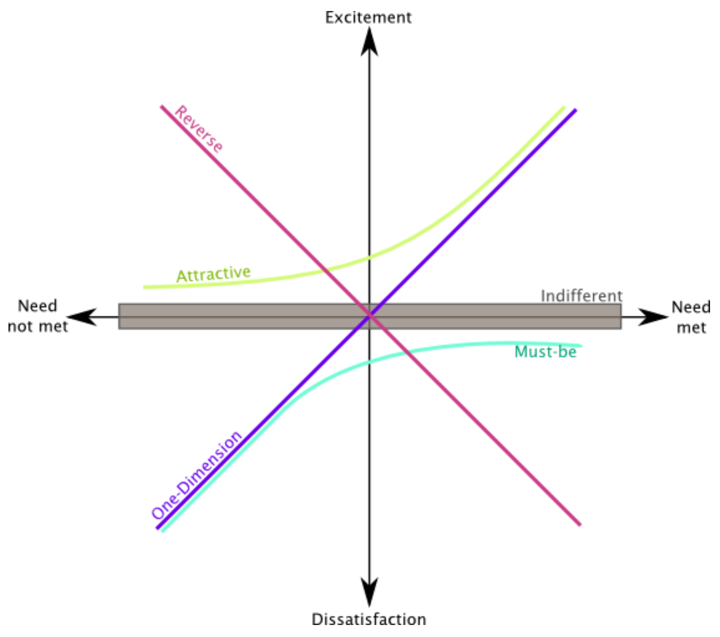
One-Dimensional - This is a requirement which when met will deliver satisfaction, however when it is not met will lead to dissatisfaction. For example if a yogurt tastes nice then it will lead to satisfaction, however a bad tasting yogurt will lead to dissatisfaction.These requirements will have high priority.

Must-be - This is a requirement which is assumed to be completed, it will lead to dissatisfaction when not delivered but cannot lead to satisfaction. These requirements will have high priority. For example, it is assumed that a yogurt pot has no holes in it.

Indifference - These are requirements which will not lead to satisfaction or dissatisfaction. These requirements will have very low requirements or be removed. For example, a user won't necessarily mind which type of recycling group the plastic falls into.

Reverse - These are requirements which lead to higher dissatisfaction the more they are met. These requirements must be removed. For example, a user may be dissatisfied with a complicated but sophisticated system even if it gives them more functionality if they wan't simplicity.

These five responses can be represented as lines on a graph (shown below) of satisfaction against level of requirement completion. ('attractive' and 'must-be' tend towards 0).



2.2 Analysis

2.2.1 The problem

Over the last 20 years Durham University Engineering department has been writing and developing a bespoke suite of software designed to operate the department's wind tunnels where each program within this suite has a specific function. There are 3 main functions undertaken by this suite; Input, Data processing and Output. To maintain compatibility, each program is written with a core "ASCII" file format readable by Tecplot for visual analysis. Currently there are over fifty programs within the suite and each program requires pre-requisite data or a specific system state to run correctly. Currently, single programs are executed from the command line using execution parameters such as I/O information and execution cycles. If necessary, sequences of programs can be executed by creating batch (.bat) files, within which a sequential series of commands are executed through the command line with no user interaction. During wind tunnel research, analysis of data may take up to 24hours, and as such the system is left unattended for long periods of time after initial execution.

Users range in experience from the software programmers themselves, to 1st Year PHD students with no experience at all. It currently takes approximately a week for a typical student to competently use the software, such that they would be able to run either a single program or execute a basic batch operation. During this learning process the students would be expected to read the large and highly detailed manual and practice executing programs without any direct supervision. Command line and batch execution is highly sensitive to syntax errors, meaning that even the most experienced of users occasionally fail to execute code due to typing errors. This problem is amplified for the inexperienced and extends the learning process.

2.2.2 User and Stakeholder Classes and Characteristics

Before we identify the stakeholders of the system it's important to establish a working definition of stakeholder:

A stakeholder is an individual or group with an interest in the success of the project be that, delivering intended results or maintaining the viability of the project. Stakeholders can influence all areas of the software project from its initial conception, to passing judgement on the final article. Stakeholders also include customers and interested members of the public (1)

Stakeholders:

1. **Durham University**

   - Durham university Engineering and Computing Sciences department is the main stakeholder of the project. Ultimately the project was conceived by the department and they provide the resources required to complete the project

2. **Engineering Department**

   - The engineering department is the client for this project. The engineering department has provided; detailed system requirements, core software for development, valuable real world feedback on how the system should work, and access to the intended hardware.

3. **The Developers**

   - The developers of the system will ultimately decided how the software will meet the requirements of the users.

4. **The Users**

   - **Experienced Users of Current System**

     i. Currently there are many users who are highly competent executing the software packages from the command line. It's important that the software package doesn't mean that these users lose functionality. Also as current users a lot of them have contributed towards the development of the core software and as such shouldn't be alienated by a simple system

   - **Inexperience Users of New System**

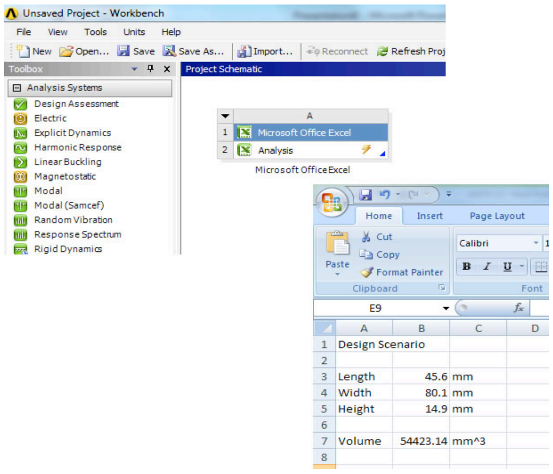
     i. New users. Currently new users struggle, however with our new system we should aim for this not to be the case as for a person to want to continue with a product then the software has to be accessible to all.

2.2.3 Solutions

Create new bespoke software – One option to solving the problem is to create a new bespoke software package. This option allows the solution to be far more tuned to the users needs. The main disadvantages to this as a solution is far more time will have to be put into the software.

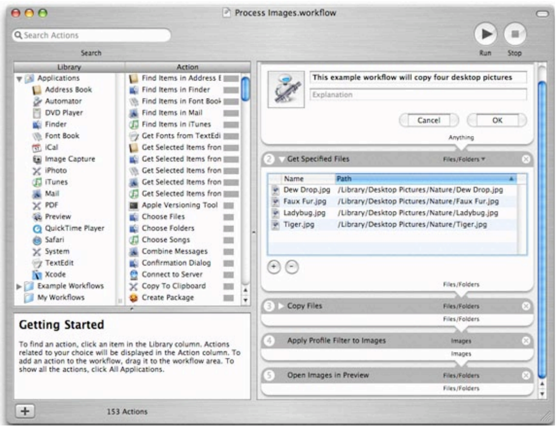
Existing Software – Whilst there aren’t any solutions designed to deal with the specific hardware and software challenges found in the Durham Engineering dept. there are however some programs that could be potentially be adapted.

ANSYS 13.0 – Ansys 13.0 is a fluid dynamics modelling package designed to enable “enable complete virtual prototyping”. Along with the ability to accurately model 3d environments and apply forces along any plane this package has potential to replace the current system. It is however only a simulation and all of the current and valuable re-sources such as the wind tunnels available to the department would not be utilised. The software also doesn’t allow adaptation so the data from the wind tunnels can’t be added.



	A	B	C	D
1	Design Scenario			
2				
3	Length	45.6	mm	
4	Width	80.1	mm	
5	Height	14.9	mm	
6				
7	Volume	54423.14	mm^3	
8				

Apple Automator – Apple Automator is a simple program that executes a series programs or actions as per the users request. Unfortunately it’s only compatible on Mac OSX and the program can’t actually input data without being told what data to input first, making it a more complicated solution than what is currently being used.



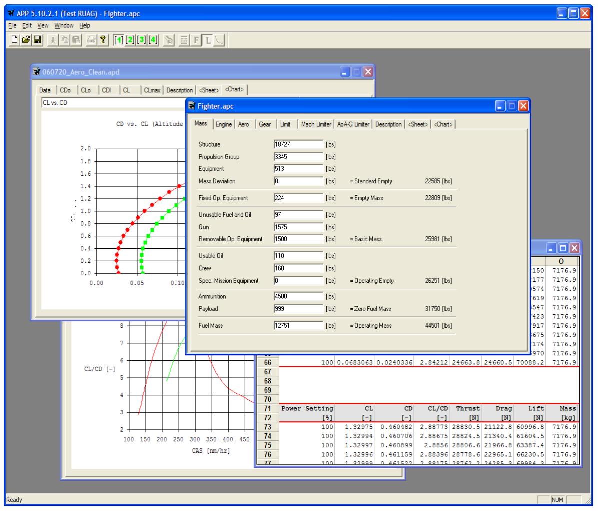
This example workflow will copy four desktop pictures

Name	Path
Dew Drop.jpg	/Library/Desktop Pictures/Nature/Dew Drop.jpg
Faux Fur.jpg	/Library/Desktop Pictures/Nature/Faux Fur.jpg
Ladybug.jpg	/Library/Desktop Pictures/Nature/Ladybug.jpg
Tiger.jpg	/Library/Desktop Pictures/Nature/Tiger.jpg

Getting Started

To find an action, click an item in the Library column. Actions related to your choice will be displayed in the Action column. To add an action to the workflow, drag it to the workflow area. To show all the actions, click All Applications.

Aircraft Performance Program - APP is a trusted aircraft performance analysis tool that designed purely for aircraft modelling. It features a high level of adaptability and is already compatible with Tecplot however it wouldn't be compatible with the software currently used by the department and there are high initial costs.


APP 5.10.2.1 (Test RUAG) - Fighter.apc
File Edit View Window Help
060720_Aero_Clean.apd
Data CDo CLo CDi CL CLmax Description <Sheet> <Chart>
CL vs. CD
CD vs. CL (Altitude
CL/CD [-]
CAS [nm/hr]
Fighter.apc
Mass Engine Aero Gear Limit Mach Limiter AoA-G Limiter Description <Sheet> <Chart>
Ready
NUM

Structure	18727	[lbs]		
Propulsion Group	3345	[lbs]		
Equipment	513	[lbs]		
Mass Deviation	0	[lbs]	= Standard Empty	22585 [lbs]
Fixed Op. Equipment	224	[lbs]	= Empty Mass	22809 [lbs]
Unusable Fuel and Oil	97	[lbs]		
Gun	1575	[lbs]		
Removable Op. Equipment	1500	[lbs]	= Basic Mass	25981 [lbs]
Usable Oil	110	[lbs]		
Crew	160	[lbs]		
Spec. Mission Equipment	0	[lbs]	= Operating Empty	26251 [lbs]
Ammunition	4500	[lbs]		
Payload	999	[lbs]	= Zero Fuel Mass	31750 [lbs]
Fuel Mass	12751	[lbs]	= Operating Mass	44501 [lbs]

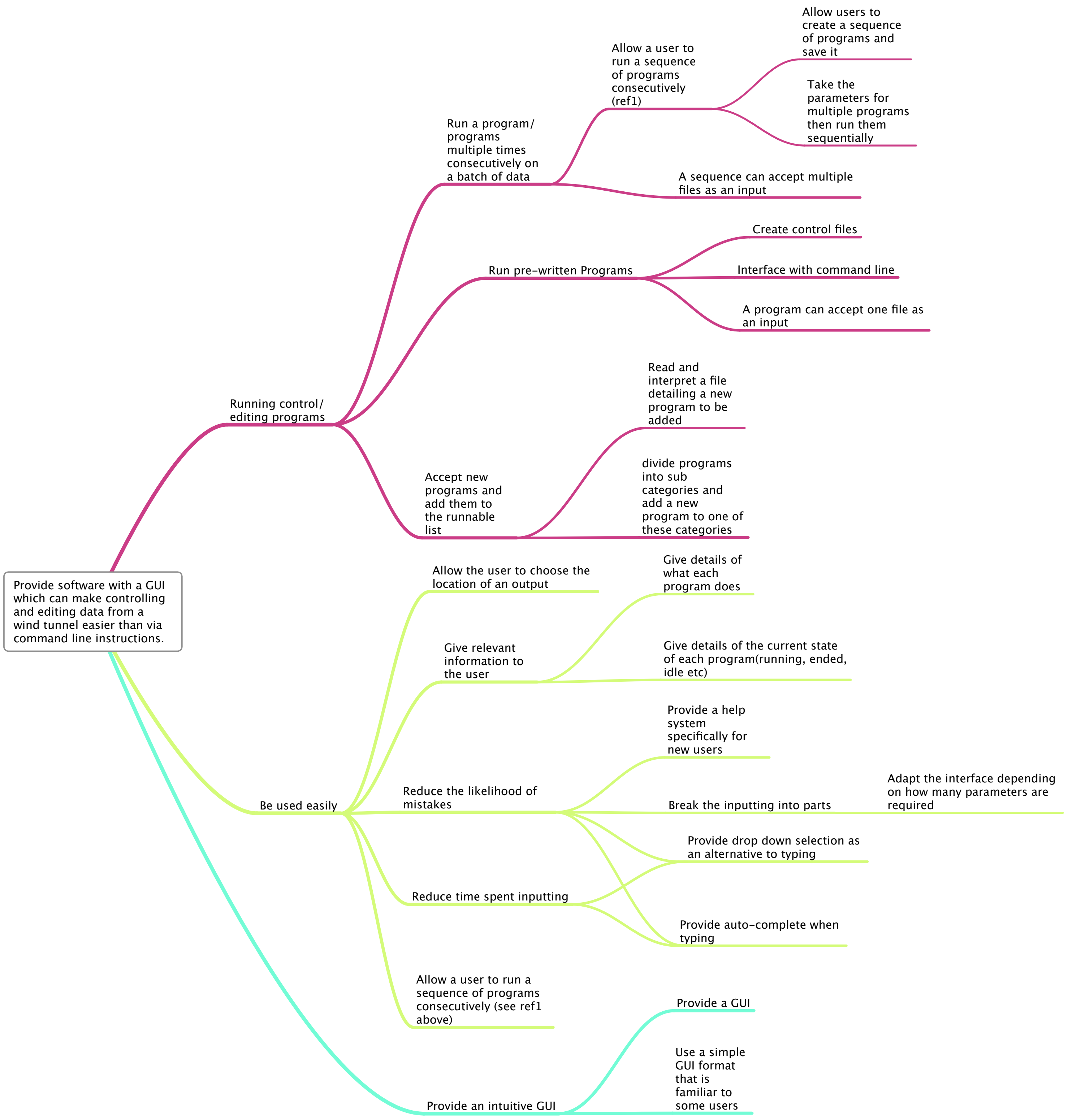
	Power Setting [%]	CL [-]	CD [-]	CL/CD [-]	Thrust [N]	Drag [N]	Lift [N]	Mass [kg]
66	100	0.0683063	0.0240336	2.84212	24663.8	24660.5	70088.2	7176.9
73	100	1.32975	0.460482	2.88773	28830.5	21122.8	60996.8	7176.9
74	100	1.32994	0.460706	2.88675	28824.5	21340.4	61604.5	7176.9
75	100	1.32997	0.460899	2.8856	28806.6	21966.8	63387.4	7176.9
76	100	1.32996	0.461159	2.88396	28778.6	22965.1	66230.5	7176.9

2.2.4 Conclusion

In conclusion whilst there are several programs available none of the programs can meet the requirements. As a result the only viable option is to create some bespoke software capable of running the pre-written programs designed by the engineering department. This software will be defined using the CTQ analysis mentioned earlier. (The CTQ tree is shown below).

2.2.3 Critical to Quality Results

The tree below details the results of extended Critical to Quality (CTQ) analysis on the given problem. The 'leaves' are the functional requirements of the project whilst the root is the project vision. Most profoundly this analysis, as well as making sure there are no excess or missed requirements, helped us to visualise the project in three major parts. Running the control programs, Ease of use and Providing a GUI.


Provide software with a GUI which can make controlling and editing data from a wind tunnel easier than via command line instructions.
Running control/ editing programs
Run a program/ programs multiple times consecutively on a batch of data
Allow a user to run a sequence of programs consecutively (ref1)
Allow users to create a sequence of programs and save it
Take the parameters for multiple programs then run them sequentially
A sequence can accept multiple files as an input
Run pre-written Programs
Create control files
Interface with command line
A program can accept one file as an input
Accept new programs and add them to the runnable list
Read and interpret a file detailing a new program to be added
divide programs into sub categories and add a new program to one of these categories
Be used easily
Allow the user to choose the location of an output
Give relevant information to the user
Give details of what each program does
Give details of the current state of each program(running, ended, idle etc)
Reduce the likelihood of mistakes
Provide a help system specifically for new users
Break the inputting into parts
Adapt the interface depending on how many parameters are required
Provide drop down selection as an alternative to typing
Reduce time spent inputting
Provide auto-complete when typing
Allow a user to run a sequence of programs consecutively (see ref1 above)
Provide an intuitive GUI
Provide a GUI
Use a simple GUI format that is familiar to some users

2.2.6 Analysis of Kano Results

The Kano Questionnaire allowed us to further analyse the features which are required by the new system. The questionnaire was answered by existing users and the average results can be seen below.
These results have been extremely useful in the prioritising of functional requirements; in that they have given us insight into those features which are both vital and those which are perhaps not as important from a user's point of view as we had first thought.

There are a multitude of features which would obviously be "must be" features, such as the creation of control files and the use of a simple GUI, which were confirmed by users via the Kano model.

On the other hand the questionnaire results demonstrated that some features would not be as attractive to the user as understood from our initial analysis. The function to accept multiple files as input may not be made use of by users- thus leaving the function with a low priority.

On-dimensional answers are the ones which are most important to us as developers, as these demonstrate when most dissatisfaction will occur if features are not included. Although some features would be thought of as one-dimensional even to a non user (allow sequential running of programs), other features which were flagged as one-dimensional have helped us to prioritise less obvious features. Adapting the interface based on the number of parameters the program requires, as well as the number of programs the user wishes to use, is a feature which we will have to make as efficient and intuitive as possible as the users' feel that is would drastically improve their use of the system.

Questionnaire for Existing Users

This questionnaire revolves around the concept of the Kano model analysis. The list below shows functions that may be included in the system and your answers will help us prioritise or eliminate potential features. The questions are answered based of a set of five criteria, as detailed below.

Attractive Attribute: This is where the feature would satisfy the user if available, however if not fulfilled there will be no repercussions caused.

One-dimensional Attribute: This measure is where the feature will give satisfaction if fulfilled but if not the user will use the system with a feeling of discontent.

Must-be Attribute: A must-be feature is one which is considered a minimum feature, which is taken for granted when present but would cause dissatisfaction if not present.

Indifferent Attribute: This is a measure of whether the user would care about the feature- whether or not it would make any difference to the program use.

Reverse Attribute: The user would expect the total opposite of this feature.

	Attractive	One-dimensional	Must-be	Indifferent	Reverse
Allow users to create a sequence of programs and save it for future use.	☐	☒	☐	☐	☐
Take the parameters for multiple programs then run them sequentially.	☐	☒	☐	☐	☐
Accept multiple files as an input.	☐	☐	☐	☒	☐
The program can accept just one file as an input.	☒	☐	☐	☐	☐
Create control files in order to run pre-written C programs.	☐	☐	☒	☐	☐
Accept new programs and add them to the list of runnable programs.	☐	☐	☒	☐	☐
Divide programs into subcategories and allow a program to be added to one of these.	☒	☐	☐	☐	☐
Give details of what each program does.	☒	☐	☐	☐	☐
Give details of the current state of each running program (i.e. running, ended, idle etc).	☐	☐	☒	☐	☐
Allow the user to specify the location of output files.	☐	☐	☒	☐	☐
Provide a help system specifically for use by new users.	☒	☐	☐	☐	☐

	Attractive	One-dimensional	Must-be	Indifferent	Reverse
Adapt the interface depending on how many parameters are required.	☐	☒	☐	☐	☐
Provide drop down selection as an alternative to typing.	☒	☐	☐	☐	☐
Provide auto-complete when typing.	☒	☐	☐	☐	☐
Provide a GUI.	☐	☐	☒	☐	☐
Use a simple GUI layout that is familiar to some users.	☒	☐	☐	☐	☐

3 Project Plan

3.1 Team Roles

Name	Role	Description
Amir Aliyu	Programming Team	This role involves working as part of a core team of three. The team will tackle the major programming problems.
Nicola Boland-Hill	Project Manager	This role involves overseeing the project as a whole. The project manager will set tasks and check that these have been done.
Craig Condron	Programming Team	This role involves working as part of a core team of three. The team will tackle the major programming problems.
Andrew Dykes	Lead Documenter	This role is focused around working on the documentation for the project. The lead documenter will compile the finished prose as well as writing a substantial part of it.
Joe Rivett	Programming Team - Designer	This role involves working as part of a core team of three. The team will tackle the major programming problems. The designer will focus primarily on the graphical aspects of the project.
Chris Salt	Team Leader	This role focuses on making sure the team is working together will. The team leader will be responsible for making sure sections are completed on time as well as making sure there are no communication issues.

Along with working on the specific areas relating to their role, all team members (excluding the project manager) are required to aid in other areas if they have time available. These ‘extra tasks’ will be allocated by either the team leader or project manager.

3.2 Deadline Matrix:

The team will conform to a set of deadlines, both implicit soft deadlines and official hard deadlines, which are laid out over the following section.

Note- rows highlighted blue indicate individual submissions

Document	Assessment type	Deadline Date	Deadline Time	Submission
Requirements Specification (soft)	Summative	30th November 2010	16:15	Techno Café- Group Meeting
Requirements Specification (hard)	Summative	3rd December 2010	17:15	DUO
Peer Assessment 1	Formative	14th December 2010	17:15	DUO
Prototype Demo	Formative	8th February 2011	16:15	Techno Café
Design Document	Summative	22nd February 2011	17:15	DUO
Implementation Demo	Summative	8th March 2011	16:15	Techno Café and on disc
Poster	Summative	3rd May 2011	17:15	Office on disc
Peer Assessment 2	Formative	3rd May 2011	17:15	DUO
Reflective Report	Summative	10th May 2011	17:15	DUO

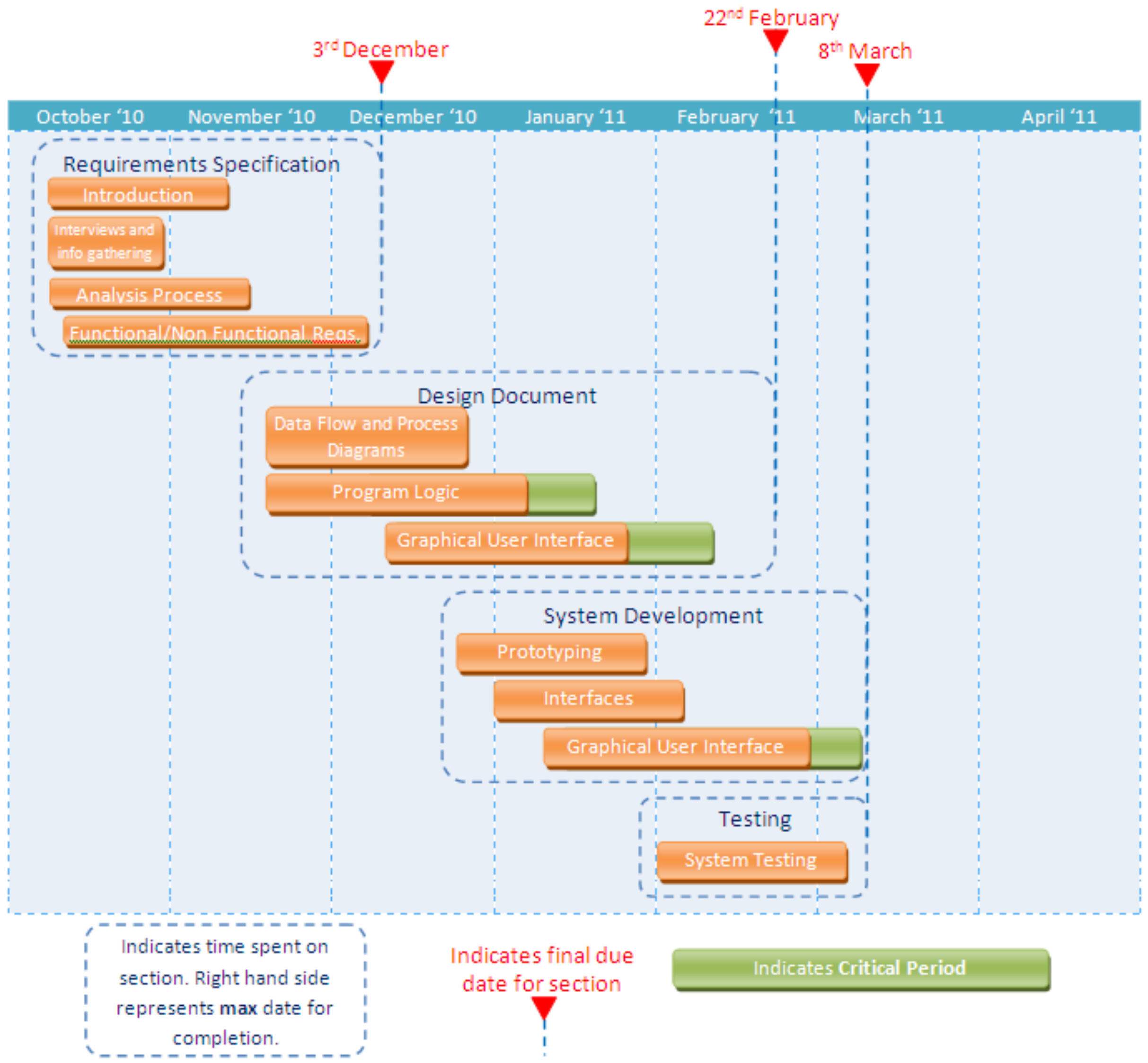
The diagram below is based on a Gantt Chart and indicates the amount of time the team will spend on each section leading up to and including the system testing. The final deadline for the implementation is the 8th March. The chart shows the time plan for each aspect of the project along with critical periods to which the team must conform.

However, there will be far more soft deadlines within the broad deadlines than those shown in the chart. For example, the design document in the chart is drastically simplified- not only in terms of general deadline headers but also in terms of soft deadlines set within the team.

There will be a multitude of other vital dates set when in the design document is under development. In order to keep to the deadline there will need to be periods designated to designing specific aspects of the Graphical User Interface, namely those that will make a vast difference to the HCI, as well as sessions in which the team programmers must design the core program onto which the GUI will be built.

The “Prototyping” period will be used to write some program features prior to the development of the full system. This will give the team and the future users of the system an opportunity to test the efficacy and functionality of the prototypes that we develop.

Note that certain sections overlap as it will be possible to work on sections even if previous sections have not been fully completed. An example of this is that as a team we will start developing the GUI for the actual system before the full GUI has been designed. This will be possible as some aspects of the front end of the system can enter initial development before other aspects have been designed. This will allow for longer time not only developing the system but also designing it.



4 Hardware and Software Platforms

4.1.1 Software Platforms To Be Used

When working on the program we will be using the Eclipse IDE with the most recent software update available (at the moment of writing this document, this is Eclipse Helios version 3.6). This is because Eclipse offers fantastic cross operating system compatibility. Furthermore, Eclipse offers subversion integration with its Subversive plug-in.

Our program will be written using Java JDK 1.6, specifically J6SE. It will therefore rely heavily on java as this is the primary programming language that we will be using (if not the only language). The reason being that like Eclipse, Java offers wide compatibility across a range of systems.

In terms of operating systems, we shall be using be using Mac OSX, Windows 7 and the Ubnuntu Linux distribution. As stated earlier this will not be a problem because of both Eclipse's and Java's cross compatibility.

We shall be using the tortoise SVN control for version control. This is because the Tortoise SVN client is both free and easy to use.

4.1.2 Hardware Platforms to be Used

For the creation of this program it is assumed all hardware platforms to be used will be of a minimum spec. All machines will be running on an intel based platform with the following minimum specifications:

- Intel Core 2 duo(1.6ghz per core)
- 1GB DDR2 Ram
- 160gb HDD

4.2.1 Software Requirements

Therefore for our program to run on the client computer/computers the JVM (Java virtual machine) must be installed.

(An internet connection would be desirable as updates to all aspects of java could be downloaded to enhance performance on the client computer; however it is not a necessity once java and the java vm is installed.?)

We have assumed that the client computer running the program is to be run on has access to the Local Durham Area Network. This is necessary because the program will be stored on the network. We feel this assumption is valid because current client computers have using the wind tunnel software have access to this network.

Now assuming the client computer can run the existing wind tunnel software, we can safely presume it will also be capable of running the java virtual machine and therefore run our own software.

The program we create must be compatible with Windows XP. This constraint arises as the existing wind tunnel software has been primarily created for XP and this is the OS installed on the computer. This is one constraint upon our software which we can easily overcome by using java, as it can be compiled to run on any OS/hardware setup once it has been programmed and recompiled if necessary.

Our program can support both 32 bit and 64 bit versions of Windows XP.

Our program also supports both Windows XP and Windows classic desktop managers.

4.2.2 Hardware Requirements

There must be enough physical storage on the client machines drives so that our software can be installed along with any files it depends on (i.e. java files, input files etc)
The computer's hardware must meet the minimum specifications that are required to install and run an up-to date version of java. (Currently as of 16th November 2010, at http://www.java.com/en/download/manual.jsp the file size of the java installer - assuming an internet/online connection is 10mb – without an internet connection, the installer is 15.3mb)

There must be at least 1GB of RAM installed on the computer/computers in the lab so that our software will run smoothly and in real time when carrying out its tasks.

Mouse and keyboard peripherals are also necessary for the users to operate our software and input required/desired data or interface with it.

The System that the program is to be run on should have at least 500mb of space free.

5 References

Author: Dr David Sims-Williams (2010). DSW Manual. Published in Durham: Durham University. 82 Pages.

Dr David Sims-Williams, Sarah Drummond: (2010). DSW Brief. Published in Durham: Durham University. 2 Pages

Colin Dykes: (2010). Lecture - Project Management MSc - Scope. GlaxoSmithKline, Project Management

Oracle: Sun/Oracle. (Published 2010). What are the system requirements for Java 6?. Available: **http://www.java.com/en/download/help/6000011000.xml**
Last accessed 22nd Nov 2010.

Wikipedia Users. (Last modified on 23 November 2010). System Requirements. Available: **http://en.wikipedia.org/wiki/System_requirements** Last accessed 23rd Nov 2010.

Wikipedia Users: (Last modified on 16 November 2010). Java Virtual Machine. Available: **http://en.wikipedia.org/wiki/Java_Virtual_Machine** Last accessed 23rd November.

Ken Black: (2010). What is a Stakeholder?. Available: **http://www.wisegeek.com/what-is-a-stakeholder.htm** Last accessed 25th Nov 2010.

Thiyagarajan Veluchamy: (2008). Glossary. Available: **http://thiyagarajan.wordpress.com/glossary/** Last accessed 25th Nov 2010.

Administrator of www.lessons-from-history.com: (2009). Functional VS NonFunctional Requirements and Testing. Available: **http://www.lessons-from-history.com/node/83**
Last accessed 30th Nov 2010.

Ansys. (2010). Ansys Product Portfolio. Available: **http://ansys.com/products/default.asp**
Last accessed 2nd December 2010

Design Analysis Research Corporation. (2010). Aircraft Performance Program. Available: **http://www.darcorp.com/Software/app/** Last accessed 2nd December 2010.

Apple Inc.. (2006). Working With Automator. Available: **http://developer.apple.com/macosx/automator.html** Last accessed 2nd December 2010.

6 Definitions

ASCII – (American Standard Code for Information Interchange) is a character encoding format used by the computer to encode the English alphabet and some symbols

Command Line – A text based method of inputting instructions to a computer system

Data Analysis – Is the breakdown and use of data to reach conclusions

Domain Analysis – Is the process of capturing current information available from related and current software in the domain with the intention of using the information in a future system

Eclipse – is a software development environment which will be used to write the program

Functional Requirement – A function which the program must perform (could be a desired input, mutation and/or output)

Java – Is a programming language which is object orientated and will be used to write our program

NonFunctional Requirement – A requirement which the program must meet, which does not relate to what the program must do, and is instead based on its efficiency, its portability, how easy it is to use, its maintainability, and how reliable it should be.

Parameter – a value that is passed into a program or sequence

Problem Domain – The area we need to examine in order to solve the specified problem and come up with a solution

Programmer – Person who writes the computer code for the programme (may also test the program aswell)

Program Definition File – A file which holds all of the parameters that are required for any future programs to be interpreted by our program

Project Mandate –The brief for the project

Stakeholder – Any party who has an interest in the project

Syntax Error – Code which a programmer has entered that the computer does not ‘understand’. Often occur due to incorrect spelling

Tecplot – A program used by the engineers to output their collected data

Tortoise SVN – is a subversion control system designed for windows machines

<u>Acronyms</u>

GUI – (Graphical User Interface) is a user interface based upon images and icons, whereby a mouse and/or keyboard is used to provide input

HCI – (Human Computer Interaction) is the study of any interactions between a human and a computer

I/O – (Input/Output) is the ways of communication between a computer and the user or another computer

JDK – (Java Development Kit) is software that allows programmers to write applications and applets in Java

RAM – (Random Access Memory) is volatile memory used by the computer which can be read from or written to instantly so that it is readily available for the computer to use

SVN – (Subversion Control) is a control system used to manage updated and archived versions of a document or code (referred to as subversions), to ensure only the newest version is edited or used

7 Solution Requirements

7.1 Functional Requirements

Type	Functional
ID	F001
Description	Allow users to create a sequence of programs and save it.
Priority	High
Pre-conditions	F008, F007
Test Input	Multiple Programs will be entered into a sequence, which will be viewed both at that point and after closure and reopening of the software.
Operations	Sequence will be stored.
Expected results	The software should show the programs sequentially both times.

Type	Functional
ID	F002
Description	Take the parameters for multiple programs then run them sequentially.
Priority	High
Pre-conditions	F003
Test Input	A sequence will be chosen given a set of ordered files. The sequence will be run.
Operations	The sequence will then be executed.
Expected results	The programs should run sequentially giving correct results.

Type	Functional
ID	F003
Description	A sequence can accept multiple files as an input.
Priority	Medium
Pre-conditions	None
Test Input	A sequence will be chosen given a set of ordered files. The sequence will be run.
Operations	The sequence will then be executed.
Expected results	The programs should run sequentially giving correct results.

Type	Functional
ID	F004
Description	Run pre-written programs (Create control files)
Priority	Very High
Pre-conditions	F007, F008
Test Input	A pre-added program will be given correct parameters and run.
Operations	The program will then be executed.
Expected results	The program will be successfully run creating a control file.

Type	Functional
ID	F005
Description	Run pre-written programs (Interface with command line)
Priority	Very High
Pre-conditions	F007, F008
Test Input	A pre-added program will be given correct parameters and run.
Operations	The program will then be executed.
Expected results	The program will be successfully run.

Type	Functional
ID	F006
Description	A program can accept one file as an input.
Priority	High
Pre-conditions	F007, F008
Test Input	A pre-added program will be given correct parameters and run.
Operations	The program will then be executed.
Expected results	The program will be successfully run with correct results.

Type	Functional
ID	F007
Description	Read and interpret a file detailing a new program to be added.
Priority	Medium
Pre-conditions	None
Test Input	A program definition file will be given to the software.
Operations	The software will interpret that file.
Expected results	A new program will be added to the list of runnable programs.

Type	Functional
ID	F008
Description	Divide programs into sub-categories and add a new program to one of these categories.
Priority	Medium
Pre-conditions	None
Test Input	A program definition file will be given to the software.
Operations	A sub-category is selected by user.
Expected results	A new program will be added to the list of runnable programs.

Type	Functional
ID	F009
Description	Allow a user to choose the location of an output.
Priority	Low
Pre-conditions	None
Test Input	Run a program specifying an output location.
Operations	Output location is specified by user in the software.
Expected results	Upon completion the output file will be in the correct location.

Type	Functional
ID	F010
Description	Give details of what each program does.
Priority	Low
Pre-conditions	F007
Test Input	Attempt to view descriptions of program
Operations	The software locates correct description on action.
Expected results	A correct description is shown.

Type	Functional
ID	F011
Description	Give details of the current state of each running program (i.e. running, ended, idle etc)
Priority	Low
Pre-conditions	None
Test Input	Run a program checking it’s state.
Operations	The software monitors the system state.
Expected results	The program will move from idle, to running, to ended at the correct times.

Type	Functional
ID	F012
Description	Provide a help system specifically for use by new users
Priority	Medium
Pre-conditions	None
Test Input	Test for availability of help system.
Operations	The software locates the correct help data, for that program.
Expected results	Help system is available.

Type	Functional
ID	F013
Description	Adapt the interface depending on how many parameters are required.
Priority	Medium
Pre-conditions	F007, F016
Test Input	Select multiple different programs which require different amounts and types of parameters.
Operations	The Software retrieves the correct number of data inputs.
Expected results	The GUI should change, giving a different amount of text boxes to enter in.

Type	Functional
ID	F014
Description	Provide drop down selection as an alternative to typing.
Priority	Low
Pre-conditions	F016
Test Input	Test for availability of drop down lists.
Operations	The software should look up all relevant data for that drop down list.
Expected results	Correct drop down lists are available.

Type	Functional
ID	F015
Description	Provide auto-complete when typing.
Priority	Low
Pre-conditions	F016
Test Input	Test for availability of auto-complete.
Operations	The software should look up all relevant data for to fill in the auto-complete.
Expected results	Correct auto-complete is available.

Type	Functional
ID	F016
Description	Provide a GUI.
Priority	Very High
Pre-conditions	None
Test Input	Test for availability of GUI.
Operations	None
Expected results	GUI is available.

Type	Functional
ID	F017
Description	Use a simple GUI format which is familiar to some users.
Priority	Medium
Pre-conditions	F016
Test Input	Ask users to familiarise themselves with the GUI.
Operations	None
Expected results	All users will meet time taken to familiarise targets.

7.2 Non-Functional Requirement

Type	Non-Functional
ID	NF001
Description	A novice user should be able to initiate one pre added program within 5 minutes of being introduced to the software
Priority	High
Pre-conditions	The system is complete
Input	A novice user with no experience with the system is given a task to perform
Operations	Novice user carries out the task
Expected results	The pre added program is successfully executed within 5 minutes.

Type	Non-Functional
ID	NF002
Description	The program should be compatible with the minimum specifications as outlined in section 4
Priority	High
Pre-conditions	Refer to section 4
Input	the program is executed on a machine with the current hardware and software specifications
Operations	A sequence already stored by the system is executed
Expected results	The sequence is successfully executed

Type	Non-Functional
ID	NF003
Description	The system must be robust against unexpected inputs
Priority	Medium
Pre-conditions	The system is complete
Input	Inputs that are contrary to what is normally expected by the system. An example would be inputting a .png file.
Operations	The system refuses to run
Expected results	The system provides an input not suitable error message

Type	Non-Functional
ID	NF004
Description	The GUI must meet at least 50% of the HCI display requirements
Priority	High
Pre-conditions	The system being complete
Input	The system's GUI
Operations	The system is being checked against the HCI requirements
Expected results	The system passes the test.

Type	Non-Functional
ID	NF005
Description	The system should remain stable during execution for up to 24hours.
Priority	High
Pre-conditions	None
Input	All pre added programs are run repeatedly in a 24 hour span
Operations	The programs execute for 24 hours
Expected results	Execution of all programs is successful and all program data is generated.

Type	Non-Functional
ID	NF006
Description	The program must be scalable, with regards to accepting large amount of input
Priority	High
Pre-conditions	F007 & F008
Test Input	A given program will be tested with double the amount of input normally entered
Operations	The program operated on the inputs
Expected results	The program successfully executes

Type	Non-Functional
ID	NF007
Description	The system must be reliable, by providing robust error handling
Priority	High
Pre-conditions	None
Test Input	An input file manufactured to create an erroneous system state.
Operations	The program fails to run
Expected results	The program produces a suitable error handling depending on the erroneous state created by the input