Durham University
School of Engineering and Computing Sciences

Requirements Document

2010-11

Project Name: Durham Software For Wind Tunnels

Team Name: Creative Solutions

Team Number: 7

Team Members:

Carl Dobson
Joseph Flynn
Christopher Miley
David Roberts
Adam Wardle

Document Information

Project Name:	Wind Tunnel Software		
Prepared By:	Creative Solutions	**Preparation Date:**	02/12/2010
Email:	cs-seg7@durham.ac.uk		
Document Version No:	4.0	**Document Version Date:**	02/12/2010

Version History

Ver. No.	Ver. Date	Revised By	Description
1.0	07/11/2010	Christopher Miley	Created requirements document.
1.1	09/11/2010	David Roberts	Fixed version numbering, Gantt chart cropping and wording of certain deadlines.
1.2	11/11/2010	Carl Dobson	Started Revising Definitions of solution requirements – Functional Requirements.
1.3	14/11/2010	Christopher Miley	Re-ordered and added new sections, added new and updated content to sections 5, 3 & 7.
2.0	15/11/2010	Christopher Miley	First draft complete.
2.1	15/11/2010	Christopher Miley	Document formatting.
2.2	18/11/2010	Christopher Miley	Partial edit, updated Section 5.
2.3	18/11/2010	David Roberts	Fixed crop line on logo and cleaned up front page and header a little.
2.4	19/11/2010	Carl Dobson	Updated Section 6 solution requirements, also updated the Definition of Terms.
2.5	19/11/2010	Adam Wardle	Fixed the misalignment of page margins in Section 4.
2.6	19/11/2010	SEG 7	Group review.
2.7	22/11/2010	Adam Wardle	Corrected a spelling mistake in Section 5, and the colour coding in the Soft Deadlines and Dependencies in Section 4. Also corrected some problems with formatting.
2.8	23/11/201	Carl Dobson	Update Solution requirements and fixed paragraph alignments.
2.9	23/11/2010	Adam Wardle	Added further definitions to Section 7.
2.9b	25/11/2010	David Roberts	Proofreading 2.9 and placing comment bubbles for required alterations.
3.0	29/11/2010	Christopher Miley	Updated Sections 1, 3 and 5. Improved the formatting of the header and footer and second draft complete.
3.1	30/11/2010	Carl Dobson	Updated reference section, also updated priorities in section 6 and also started to update section 5.
3.2	30/11/2010	Adam Wardle	Updated section 2, and the references for section 2.
3.3	30/11/2010	David Roberts	All of the corrections from the meeting (30/11/2010).
3.4	02/12/2010	Carl Dobson	Added an extra solution requirement for creating a log of operations.
3.5	02/12/2010	Christopher Miley	Updated page numbers, added further references and definitions of terms, updated Section 5 and fixed formatting issues.
3.6	02/12/2010	Adam Wardle	Minor formatting corrections throughout document.
3.7	02/12/2010	Carl Dobson	Fixed footer as was not in printable area.
3.8	02/12/2010	Christopher Miley	Proof read, updated formatting and fixed mistakes in definitions of terms.
3.9	02/12/2010	Adam Wardle	Corrected some punctuation in section 6.
4.0	02/12/2010	SEG 7	Final version.

Table of Contents

1. Purpose (Executive summary)

Creative Solutions has been approached by the Engineering Department within the University of Durham, to develop a system that will provide a graphical user interface (GUI) and management front-end to control an already developed suite of command line based applications. These applications manage and analyse data output from various wind tunnels that are owned by the Department. The suite is used in over ten facilities for both real time and subsequent data analysis. The new system must reduce the steep learning curve end-users currently face when trying to use the system.

Within this document are the outlines for the project. These include, but are not limited to; domain analysis, project plan, solution requirements and hardware and software platforms. This document should be referred back to and modified as required throughout the design and development process.

2. Domain Analysis

2.1 Analysis Process

In order to create a GUI for the Durham Software for Wind Tunnels suite, it is essential that the existing system is fully understood. The process by which this was achieved is as follows.

The analysis of the existing system began by reading the Durham Software for Wind Tunnels manual. It gives details of how to install and run the software. It lists all of the programs available within the suite, along with their required inputs, outputs and a description of what the program does. This gave us an initial idea of how the existing system works and what the programs are used for.

In order to increase our understanding of the existing system, the customer gave us a brief tour of the wind tunnel facilities. This allowed us to see the equipment and existing software in action. Following the tour, the customer gave us a quick demonstration of how to use the software, including running the programs and how to do batch processing. This raised many questions, which were then posed to the customer during an interview. This allowed us to clarify anything we did not understand about the system, and to ask him his opinions on how the system could be improved.

We also had interviews with two of the current system's users; one who was a relatively new user, with around one year of experience and the other who was an experienced user, with about five years of experience. This helped us to identify their usability issues with the existing system, and how the system could be improved to solve these issues.

2.2 Analysis

Business Environment

The main purpose of the Durham Software for Wind Tunnels suite is to enable users to gather and process data from the Engineering Department's wind tunnels. It provides them with the necessary tools to enable them to carry out experiments in the wind tunnels, and process the data they collect. This data is then used to draw conclusions from the experiment.

Existing System

The existing system comprises of around 80 programs, which are split into 3 main categories; probe calibration, data logging and data processing. All input and output files for the programs are space delimited text (ASCII) files, with various extensions. This allows them to be opened by many different third-party software packages, such as Notepad or Microsoft Excel, allowing the user to view the data in different ways.

The probe calibration programs are the starting point for carrying out most tasks. They are used to calibrate the equipment used, and they produce an output in the .in format, which is then used as an input for the other programs. The data logging programs are then used to record the data gathered from the equipment during the experiments. Some of these programs produce outputs in the Tecplot ASCII point format (.tec), which is used for visualising the data using the Tecplot program. This format can also be imported into Microsoft Excel, should the user choose to do so. The data processing programs are used to process the data

that has been collected. Most of these programs produce outputs in the .tec format.

The programs are often run in particular sequences, which can sometimes take several hours to complete. There is batch processing support to help with this. However, should an error occur during the execution, the program will terminate. This can be very annoying for the user if it occurs during lengthy tasks.

All of the existing software is run from a command line. This can be especially difficult for new users, as they often have little or no experience of using a command line. The programs also don't provide the user with any on-screen help. This means that they need to read though the manual to learn how to use each of the programs they need, which can be quite time consuming.

The ASCII file inputs for the programs are often edited by the user with Notepad, and can therefore be prone to errors. Users have said that the error messages provided by the system are often quite vague and unhelpful, making it difficult for the user to find and correct the problem.

The programs are currently stored centrally on the W:\ drive. The users have informed us that it can often go down, so they tend to keep copies of the programs saved locally, to ensure that they won't find themselves unable to access the software. However, this can lead to users not running the most up to date versions of the programs.

Stakeholders

Stakeholders are any groups of people who have an interest in the Durham Software for Wind Tunnels suite. It is imperative that the different needs of these groups are taken into consideration when designing the GUI for the system. The stakeholders are as follows.

End Users – The users of the existing system include students and researchers. Students often use the department's wind tunnels to conduct experiments as part of their final year project. Researchers also use the wind tunnels to carry out the necessary experiments for their research. Both of these groups must therefore be able to use the software with ease, to allow them to collect and process data they require. At present, this is often not the case, due to the complexity of using the existing system.

Engineering Staff – The Engineering Department's staff developed the system, and they are responsible for maintaining the software. They also develop new programs as and when required and typically they develop around three or four new programs per year. Therefore it is important that new programs can be added to the GUI without the need to alter the source code. The staff may also have to help their students if they encounter any problems with the software, which they cannot resolve themselves with the help of the manual. This means that it is essential that the staff have a thorough understanding of how to use the system.

The Engineering Department – The existing suite of programs belong to the University's Engineering Department, and it is their students and staff that use, develop and maintain the system. The department also benefits from the knowledge gained through research carried out with the help of the existing system. As a result, all the functionality of the existing system must be maintained.

Durham University – The University provides its students with research-led teaching. It is therefore vital that

the Engineering Department has all the necessary tools to carry out its own research. The existing software is one of the tools used.

External Companies – The department sometimes carries out research on behalf of external companies, and reports back with their findings. The data produced must be in a format which can be understood and used by the company, so Tecplot is typically used to produce visualisations of the data. Consequently, the current support for Tecplot must remain in place.

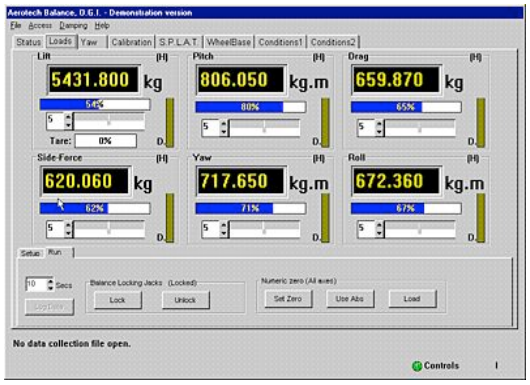
3. Competitor's Software

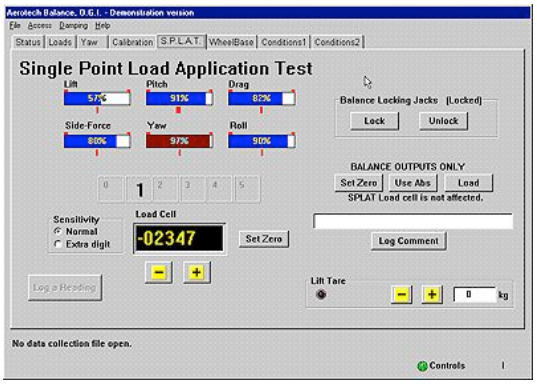
Aerotech

Developer: In house team / Platform: Windows / Website: http://www.aerotech-ate.com

Aerotech manufacture and maintain wind tunnels. Much like the Durham system, they found it best to develop their own custom suite for their equipment. The software has the ability to control the equipment.

The system is modular so that components can be added or changed easily, much like the current Durham system. Where it differs is that it has been developed from the ground up with an inbuilt GUI. It is designed to work solely on the Windows platform. While this software is a lot more complicated than what we will deliver, as it controls the wind tunnels, we can look at its layout to help us with our GUI where needed.

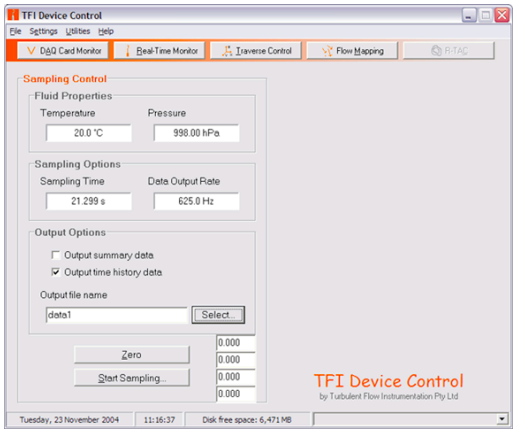


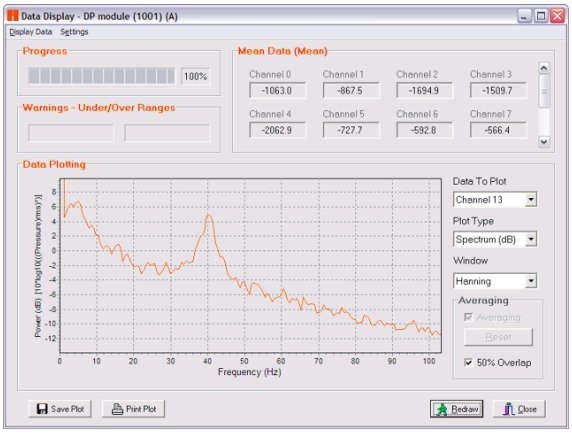


Turbulent Flow

Developer: In house team / Platform: Windows / Website: http://www.turbulentflow.com.au

Turbulent Flow, developed by an Australian software house, is similar to what we will develop for the Durham system. It collects data from external probes both in real time and for post analysis. It allows input from multiple sources and has graphing facilities built in.





4. Project Plan

Responsibilities

In order to better determine the tasks to be carried out, and in order to delegate them, roles have been loosely defined within the team by using a responsibility matrix as follows:

Team member	Implementation responsibility	Documentation responsibility	Testing responsibility	Miscellaneous
Adam Wardle	Large	Medium	Shared	N/A
Carl Dobson	Large	Medium	Shared	N/A
David Roberts	Large	Medium	Shared	N/A
Joseph Flynn	Medium	Large	Shared	Time commitment to other concurrent projects
Chris Miley	Medium	Large	Shared	Time commitment to other concurrent projects

This matrix is in place to allow dabbling between different areas – whilst Adam, David and Carl are more interested in implementation than Chris and Joseph, Chris and Joseph still want to play a role in implementation – and vice versa for documentation. Using this matrix, members with a larger responsibility in an area will coordinate and delegate tasks to the others – but overall the burden will be shared. As such there are no fixed dictatorial roles within the team, only varying levels of interest and responsibility.

Fixed Deadlines (Deliverables)

The following dates are fixed in time and purpose, except for a soft completion date (i.e. the team will strive to reach the deadline at the soft date ready for the *real* date, so there is slack prior to the deadline for last minute preparations). Note that these soft deadlines may need to be refined later in the project to reflect workload (especially those with N/A – it is too soon to decide exact deadlines).

Date	Purpose	Soft completion date	Priority	Notes
03/12/2010	Requirements Spec.	30/11/2010	High	
14/12/2010	Peer assessment	N/A	High	Individual
08/02/2011	Prototype Demo	01/02/2011	Medium	Formative
22/02/2011	Design Document	N/A	High	

08/03/2011	Implementation Demo	01/03/2011	Important	
03/05/2011	Poster	N/A	Unknown	
03/05/2011	Phoenix Presentation	N/A		Concurrent project deadline
03/05/2011	Peer assessment	N/A	Important	
10/05/2011	Reflective Report	N/A	Important	

Soft Deadlines and Dependencies

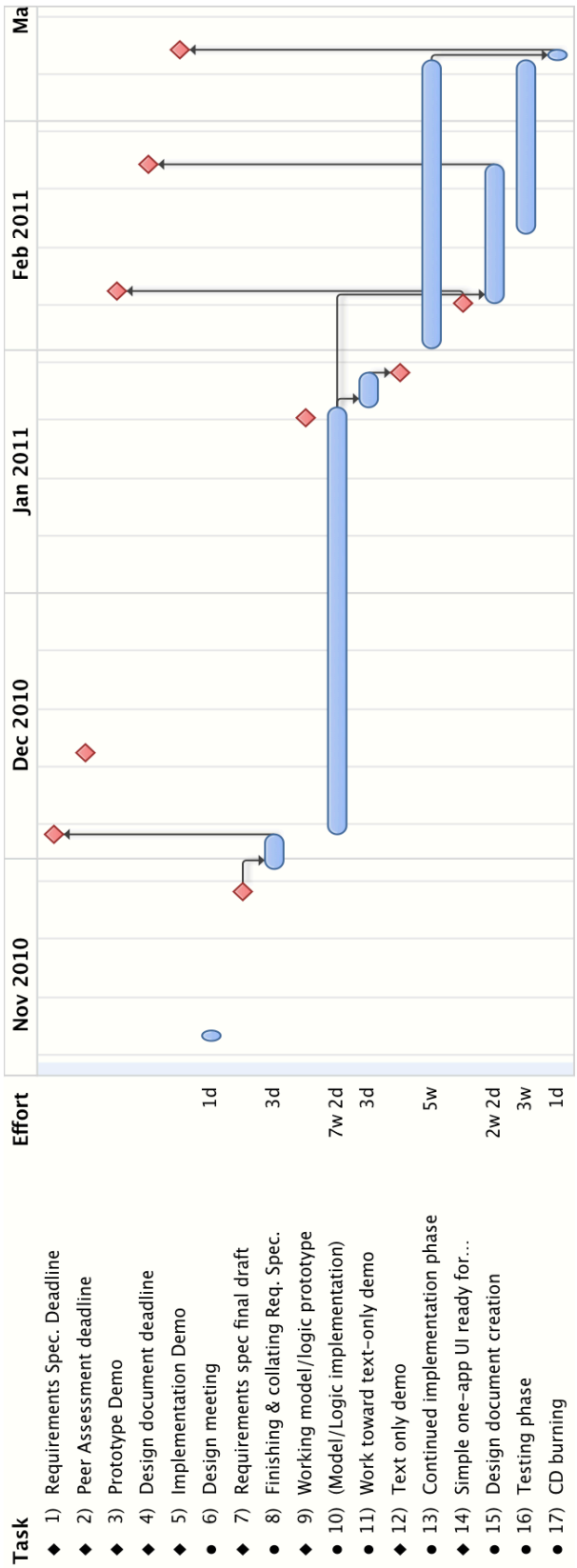
The following team-decided (soft) deadlines are in place in order to implement the system. Note that for a mixed hard/soft deadline view; see the Gantt chart provided.

Date	Purpose	Priority	Allocated to	Type	Notes
02/11/2010	Requirements Spec Draft	High	Team	Documentation	
04/11/2010	Requirements Spec Iteration 2	High	Team	Documentation	
09/11/2010	Design meeting	Low	Team	Technical Design	Code design etc. Specifically between Carl, Adam and David
26/11/2010	Requirements Spec Final draft	High	Team	Documentation	
30/11/2010	Finalising + collating Requirements Spec	High	Team	Documentation	Chris in charge of collation
24/01/2011	Working model/logic prototype (Tecplot parsing, program definition parsing, on-the-fly GUI definition generation, command line interface, logging + error interception)	High	Team	Implementation	**David**: Program definition parsing and Tecplot parsing **Carl**: In file generation and workflow **Adam**: GUI definition generation **Joseph**: Command line interface **Chris**: GUI design/ logging system
27/01/2011	Working text-only demo (logic layer working)	Medium	Adam, Carl & David	Implementation	

01/02/2011 – 07/03/2011	Continued implementation phase (overlaps testing)	High	Team	Implementation	
06/02/2011	Simple one-app interface working ready for demo	Low	Team	Implementation	The demo is formative and not a 'hard' deadline
07/02/2011 – 22/02/2011	Design document creation	High	Chris & Joseph	Documentation	**Chris** and **Joseph** coordinate: whole team contributes
15/02/2011 – 07/03/2011	Testing	High	Team	Implementation	Whole team will test as implementation progresses
07/03/2011	CD burning of final product	High	TBA	Release engineering	Who is in charge of release engineering and burning the final CD will need to be decided closer to the time

Dependencies

- The deliverable requirements specification requires a final draft to have been completed and proofread.
- The working model depends upon completion of the final requirements specification.
- The design document depends upon the completion of a simple logic prototype (the prototype will validate technical assumptions or highlight any 'course corrections' required).
- The working text-only demo, along with the one-app interface demo, requires a working logic prototype (the GUI definition generator is especially important for the interface demo).
- The demo deadline requires the working one-app demo to be complete.
- The testing requires at least some implementation in order to begin, as well as a finalised requirements specification + design document to test against.
- CD burning requires implementation to have finished successfully.


Task
Effort
Nov 2010
Dec 2010
Jan 2011
Feb 2011
Ma
1) Requirements Spec. Deadline
2) Peer Assessment deadline
3) Prototype Demo
4) Design document deadline
5) Implementation Demo
6) Design meeting
1d
7) Requirements spec final draft
3d
8) Finishing & collating Req. Spec.
9) Working model/logic prototype
7w 2d
10) (Model/Logic implementation)
3d
11) Work toward text-only demo
12) Text only demo
5w
13) Continued implementation phase
14) Simple one-app UI ready for...
15) Design document creation
2w 2d
16) Testing phase
3w
17) CD burning
1d

5. Hardware and Software Platforms

5.1 Hardware Platforms

There are a number of hardware platforms that our system must be able to work on. Our system must also be able to satisfy any requirements in regards to the hardware platforms on which it will run; this section of our document details this.

The primary hardware platform that the system must be able to operate on is a standard networked Durham Engineering Department computer. Along with this, the Engineering Department use the W:\ drive (see definitions) to work on. Both the client and users of the system mentioned that the W:\ drive is unreliable and often users store files locally. Therefore, storing the files locally is the only option for a stable working environment. The users we interviewed also stated that it would be beneficial to build local copying of files (from the W:\ drive to local storage) into the new system, to minimize the possibility of losing data. This would then eliminate any potential issue if their data or files were to not save properly on the W:\ drive, as the system would be able to provide them a backup. It would also be beneficial if our new system could automatically keep some local files (such as the wind tunnel applications) synchronized with files from the W:\ drive.

A further hardware platform that was highlighted was the fact that the probe calibration is only possible on one computer in the Engineering Department. Therefore, it is imperative that our new system must be able to work on this one computer and any other machine intended to work with the wind tunnel.

5.2 Software Platforms

The software platforms that the system must conform to are as follows:

- The customer has stated that the current modularity of the different programs is to remain; this will ensure that the program suites continue to work effectively with one another. This will allow the Engineering Department to add new programs to the system without changing the code of any of the other current programs. The functionality in the current system must be carried over to the new system.

- The client noted that it would be useful, but not essential, if the GUI was to be platform independent; this would allow the system to work on multiple Operating Systems, whilst not losing any functionality between them.

- It was also highlighted that the client wishes for the core files of the new system to be able to be changed without removing any functionality. The user must be able to make changes to the system without altering the front-end of the system that we are building. Therefore the GUI must be able to adapt to changes and improvements made to the core system files.

- During our meeting with the client, it was discussed that the current system works on both Windows

and Mac OSX. However, full functionality is only available if using Windows 7 or Windows XP. This has led us to concentrate our efforts on compatibility with Windows (7 or XP) above all else.

- The current system outputs data in an ASCII-based format designed for an application called Tecplot. The use of this format gives the department several advantages – most notably the format is interchangeable with reporting and charting tools such as Microsoft Excel. With Excel, users can transform raw data from Tecplot files into charts and diagrams. Therefore, to support the workflow of the department and to allow the continued creation of charts and reports, our system must continue to use the Tecplot format. On top of this, we must not remove the Tecplot-related functionality from the existing wind tunnel applications.

- The current system's input and output files are formatted as space-delimited ASCII, edited by most users in Notepad. Whilst our system must create a more usable interface to the wind tunnel applications, it must not prevent more experienced users from continuing to manually write and execute these ASCII files without the aid of a frontend.

- The wind tunnel applications report and return error messages in a very specific format when problems are encountered with the input files. Because of this, our system must extract and display these error messages when appropriate – and if possible, provide some extra information. This is because the primary complaint from users we interviewed was that the current wind tunnel applications generate vague and meaningless error messages.

5.3 Hardware and Software the user may need

In order for our system to function as required, the user may need the following hardware and software:

- The Java Runtime Environment (JRE) will need to be installed on the user's machine, in order to run the system because the development tool of choice is Java.

- The following specifications are the Durham University ITS minimum specifications for all machines (PC's) on the network; Processor: Intel P4 3.0Ghz, Memory: 1024Mbytes (1Gb), Hard disk: 40 GB, Video Resolution: 1024 x 768 , Supported network card: Intel pro family.

- The minimum machine specifications to run the JRE are as follows. The machines used to run our system must also satisfy the following as well as the ITS requirements above;

  Windows 7 – Memory: 128MB, Browser: IE8 or Firefox, Free Hard Disk Space: 98MB.
  Windows XP – Memory: 128MB, Browser: IE6 Support Pack 1, IE7, IE8 or Firefox, Free Hard Disk Space: 98MB.

- The user may wish to store files locally rather than on the networked W drive (as discussed in hardware platforms) so a recommended storage size of 10GB free space should be sufficient.

- The user may require a printer, in the event that reports generated by the system are to be printed.

6. Solution Requirements

6.1 Functional Requirements

6.1.1 System Functional Requirements

Name:	6.1.1.1 Minimum of 20 working programs
Description:	The system must have a minimum of 20 working programs in its suite. As per the users requirements in the design brief for the system.
Priority:	High
Pre Condition:	The executable 20 programs
Input:	Select program in the system
Operations:	System will then execute program
Expected Output:	Programs will execute successfully

Name:	6.1.1.2 Add a new Program
Description:	The system should be able to add new programs that execute in a similar style, i.e. through the command prompt using control files. This will be achieved when given a simple text file (created by the executable's maintainer), which defines how the program works; we will generate a GUI on the fly and interact with the program.
Priority:	High
Pre Condition:	New Executable and ASCII text file defining input variables
Input:	Executable and input definition file
Operations:	System will store data
Expected Output:	New program will be selectable, with a GUI for editing of the input file

Name:	6.1.1.3 Delete a Program
Description:	The system should have the ability to delete an old executable program that is obsolete and is no longer used. The program file should not simply be deleted – this is in case the program is repurposed in the future. Therefore, the system should ask the user if they wish to remove the executable from the hard drive or not. With a simple button selection of 'move to recycle bin', 'keep file on hard drive' and 'cancel'.
Priority:	High
Pre Condition:	Program must already be loaded into system
Input:	Select to delete
Operations:	User will be asked to confirm type of delete
Expected Output:	Relative action will be taken depending on users' selection

Name:	6.1.1.4 Modify a Program
Description:	The time may come when a program, which is already loaded into the system, gets moved to a different directory. Therefore the system should be able to point the program to the new location. Also the executable code could be changed, and the definition file (the description of how the executable works) may be updated accordingly. The system must tolerate these definition files being updated, something that could occur potentially days or months after the program was first added to the system.
Priority:	High
Pre Condition:	Change in current executable
Input:	Changes made to executable and/or input definition file
Operations:	System will update
Expected Output:	Program will work with any new variables available to change

Name:	6.1.1.5 Store programs
Description:	The system must be able to save the programs that are loaded in, so that they can be used next time the system is run without the need to re enter them into the system again.
Priority:	High
Pre Condition:	Programs loaded into system
Input:	User clicks to save or exit
Operations:	System will then attempt to save
Expected Output:	Save will either complete or an exception will be thrown to let the user know that the save has failed

Name:	6.1.1.6 Retrieve/select a loaded program
Description:	The user needs to be able to retrieve/select programs that are loaded into the system, so that they can specify which programs they wish to use.
Priority:	High
Pre Condition:	Program loaded in
Input:	User selects program from list
Operations:	Program is loaded into work space
Expected Output:	User will be able to edit variables and run that program

Name:	6.1.1.7 Create new Workflow
Description:	User should be able to select a number of programs and select the order they run, as well as the number of input/control files to run the programs on. This means the user can batch process their work.
Priority:	Medium
Pre Condition:	Programs loaded into system
Input:	Select a number of programs and control files
Operations:	System will execute the programs in-order on the control file(s)
Expected Output:	All of the programs will execute, on all of the control file(s)

Name:	6.1.1.8 Add new program to a workflow
Description:	The user should be able to add programs into their workflow, using a simple selection from a list of programs that are loaded into the system.
Priority:	Medium
Pre Condition:	Program in the system
Input:	User selects program from list
Operations:	System loads the program into the workflow
Expected Output:	Program will be in the list of programs to be executed in the workflow

Name:	6.1.1.9 Modify program execution order in workflow
Description:	The user should be able to select a program in the workflow and then change the order of the execution by moving it up or down the list.
Priority:	Medium
Pre Condition:	A list of execution programs of two or more
Input:	User selects and clicks to move the program in the list
Operations:	System will swap the location of the programs
Expected Output:	The workspace execution order will adjust appropriately

Name:	6.1.1.10 Remove a program from a workflow
Description:	Users should be able to remove a program that they have added into the workflow, either by accident, or if they decide they no longer need to run that program.
Priority:	Medium
Pre Condition:	Program loaded into execution list
Input:	User selects and clicks to remove
Operations:	The system will remove the program from the list and re-order the rest of the programs in order to remove the gap
Expected Output:	The execution list will be updated

Name:	6.1.1.11 Add a new control file for programs to be executed on
Description:	The system should allow the user to create workflows with not just multiple programs, but also with multiple control files for the programs to be run on.
Priority:	Medium
Pre Condition:	A workflow
Input:	User selects a new control file
Operations:	System will load in the new file
Expected Output:	System will show the new file in files to be processed list

Name:	6.1.1.12 Modify control file execution order
Description:	User should be able to move and change the control file execution order, as they may want one file to be processed before another.
Priority:	Medium
Pre Condition:	Two or more control files loaded into files to be executed
Input:	User selects control file and clicks to move control file
Operations:	System will swap control file order
Expected Output:	Files to be executed order will be changed

Name:	6.1.1.13 Remove control file from workflow
Description:	Users should be able to remove a control file from the files to be executed in their workflow, as the user may accidently add a wrong file or may wish just to do that process at a later day.
Priority:	Medium
Pre Condition:	Control file loaded into workflow
Input:	User selects to remove
Operations:	System will remove the file and re-order the files to remove the gap in execution
Expected Output:	Files to be executed will be updated

Name:	6.1.1.14 Template Workflows
Description:	The system should come with some pre-defined workflows, which will be based on the case studies in the user guide for the current system.
Priority:	Medium
Pre Condition:	Programs in case study working in system
Input:	User will load in a template from templates folder
Operations:	The system will load in the template file
Expected Output:	All of the programs in the template file will be loaded in order into the workspace

Name:	6.1.1.15 Store workflow
Description:	Users should be able to save user defined workflows, so that they can then use them at a later date, on other projects. The user should also be given the option as to where to save the workflow so they can save it locally, otherwise they can save it on the network drive in a templates folder that the system will create as a default template location. The system will load up a file browsing tool so that the user can browse to the save directory and input a relevant name and then click save to confirm location and name of the workflow.
Priority:	Medium
Pre Condition:	Newly created workflow and security access to the save location for writing
Input:	Click save
Operations:	System will save the workflow
Expected Output:	Workflow will be saved in a folder with a name the user provides

Name:	6.1.1.16 Delete Workflow
Description:	The system should have the ability to delete an old workflow that is no longer needed. This is so the user can manage the number of workflows in case there are a huge number of redundant workflows. The system will confirm the deletion of the workflow after the user has selected to delete it in case they click on the wrong workflow by accident.
Priority:	Medium
Pre Condition:	Workflow and security access to save location
Input:	Select to delete
Operations:	System will delete workflow
Expected Output:	Workflow will no longer be on hard drive

Name:	6.1.1.17 Load Workflow
Description:	The system should have the functionality. To browse for a pre-saved workflow, either created by us or by the end user. The selection of the workflow should be done with a file chooser to simplify searching for the file rather than asking the user to type a directory address in the system.
Priority:	Medium
Pre Condition:	Workflow saved on hard drive
Input:	User selects workflow
Operations:	System loads workflow
Expected Output:	Workflow will be loaded into workspace

Name:	6.1.1.18 Create ASCII input definition file
Description:	As mentioned above, an ASCII input definition file is going to be used to define the variables used for creating the GUI. When a member of the engineering department creates a new wind tunnel executable, they must create the definition file for it in order to describe the workings of the executable to our system. To simplify this process we will provide an integrated editor to assist in generating the definition files.
Priority:	Low
Pre Condition:	New Program, knowledge of variables needed and type
Input:	User selects the name and type of variables needed for there program
Operations:	System will create a input definition file based of user selection
Expected Output:	A new file will be produced in the programs directory describing the variables in order to create a GUI from

Name:	6.1.1.19 Read input definition file
Description:	The system requires a parser to be created that can read the input definition file, in order to create the GUI for a given program.
Priority:	Medium
Pre Condition:	Input definition file
Input:	User will load in program
Operations:	System will read the input definition file
Expected Output:	System will produce a GUI so user can edit the input file easily

Name:	6.1.1.20 Progress bar
Description:	The system should give the user some feedback as to the progress that has been made through a workflow. This is likely to take the form of displaying how many individual tasks have completed in relation to the total number queued for execution. E.g. Process 4 out of 10.
Priority:	Low
Pre Condition:	Knowledge of the number of programs and number of input files in order to work out how many processes there are to be executed
Input:	Users workflow
Operations:	System will work out the amount of processes and what process number is currently running
Expected Output:	System will print out useful information for the user

Name:	6.1.1.21 Executable files to be copied to local hard drive
Description:	Programs in a workflow are to be copied onto a user-selected drive. When we were speaking with the engineers they said that the network often drops and if they are working on a project which takes hours e.g. overnight they do not want the process to have been halted because the network had dropped and the system had lost connection with the programs.
Priority:	Medium
Pre Condition:	Access to programs on network
Input:	User selects to run workflow/project
Operations:	System copies programs to project directory before execution starts
Expected Output:	A local copy of the programs (on the selected drive) removes the dependency on the network connectivity and ensures long workflows (batch processes) will complete without problems caused by remote services (e.g. networked shared drives)

Name:	6.1.1.22 Delete executable from project folder
Description:	After execution of a project (workflow), the executables that were copied to a local drive will then be removed from the local drive. This is to prevent the executables being stored permanently and consuming valuable disk space (an issue especially for students of the engineering department, since the executables would be stored on their roaming profile and would need to be synchronized to every computer they logged onto). The system will also place a text file detailing the programs that were used in this execution and the order in which they were used as a form of reference for the user. Also, if desired, this information can be shared with other users (in case they wish to recreate the execution also).
Priority:	Medium if 6.1.1.21 is fulfilled, else Low
Pre Condition:	6.1.1.21
Input:	System finishes execution of project
Operations:	The system will then delete the executable files from the users project folder
Expected Output:	The project folder will no longer store any of the executable files

Name:	6.1.1.23 Load Command Prompt into system for execution
Description:	The system needs to be able to load and pass parameters to the command prompt in order to execute the programs.
Priority:	High
Pre Condition:	Knowledge of location of command prompt, System files on windows, Utilities on Mac (In case we get the time to make the system OS compatible with Mac)
Input:	Program will load a external process, of command prompt
Operations:	System will load up command prompt
Expected Output:	System will be able to pass parameters to a command prompt

Name:	6.1.1.24 Echo command prompt
Description:	The user might want to be able to see the data that the current executable is producing in a window within the system. This is so that they can check and make sure that the program is executing correctly and the way they were expecting.
Priority:	Medium
Pre Condition:	6.1.1.23
Input:	Parameters to command prompt
Operations:	Command prompt will produce output based off the parameters
Expected Output:	System will echo this output into a window in the system for the user

Name:	6.1.1.24 Executable Paths are to be stored in the system
Description:	The global path to the executable does not need to be set. This will simplify the usage of the programs for the end user as they will not need to edit the autoexec.bat or any other variables controlling directories.
Priority:	High
Pre Condition:	Knowledge of location of executable
Input:	Program definition file will store the location of the executable
Operations:	N/A
Expected Output:	Program definition files will have the executable location stored with in them

Name:	6.1.1.25 Log files of system operations
Description:	The system should create and store local log files of what has been done on the system so that users can reference back to any errors from a previous project if need be.
Priority:	Low
Pre Condition:	None
Input:	User initiates an operation
Operations:	Operation is executed
Expected Output:	System will log information about what is executed and when

6.2 Non-Functional Requirements

6.2.1 Non-Functional System Requirements

Name:	6.2.1.1 Global location of executable files on the W:\ Drive
Description:	The executable files are to be stored globally on the W:\ drive so that every computer in the suite can get access.
Priority:	High
Pre Condition:	None
Input:	User puts executable file on server
Operations:	Computer will store executable file
Expected Output:	The executable will be available on all PC in suite through the W:\ drive

Name:	6.2.1.2 File format
Description:	The files that are used by the system must be ASCII and must work in Tecplot and Microsoft Excel as the user reads the files with these two programs in order to create a graphical output of the data.
Priority:	High
Pre Condition:	None
Input:	ASCII file
Operations:	System will edit file during execution
Expected Output:	Output file will still be in ASCII and will work in Tecplot and Excel

Name:	6.2.1.5 Reliability
Description:	All functions in the system should consistently produce the same output given the same input.
Priority:	High
Pre Condition:	None
Input:	N/A
Operations:	N/A
Expected Output:	N/A

6.2.2 Non-Functional User Requirements

Name:	6.2.2.1 Usability
Description:	The user should be able to set up and run a new workflow with (4-5 programs) within 10 minutes.
Priority:	High
Pre Condition:	None
Input:	N/A
Operations:	N/A
Expected Output:	N/A

Name:	6.2.2.2 Accessibility
Description:	The System should have accessibility features to help users use the system efficiently.
Priority:	Low
Pre Condition:	None
Input:	N/A
Operations:	N/A
Expected Output:	N/A

Name:	6.2.2.3 Documentation
Description:	The client requires documentation of the system to help them to use it. This will follow the form of documentation made while designing and creating the System.
Priority:	High
Pre Condition:	None
Input:	N/A
Operations:	N/A
Expected Output:	N/A

Name:	6.2.2.4 User guide
Description:	A comprehensive and clear user guide to be created in order to explain to the user how to use the system.
Priority:	Medium
Pre Condition:	None
Input:	N/A
Operations:	N/A
Expected Output:	N/A

Name:	6.2.2.5 Help tips
Description:	On screen help tips in the system, which will display helpful information to the user about how to use the current function they are looking at.
Priority:	Low
Pre Condition:	None
Input:	N/A
Operations:	N/A
Expected Output:	N/As

6.3 Domain Requirements

Name:	6.3.1 Area of operation
Description:	The system will be used in a Wind tunnel lab and will be used to log and process data gathered from the lab.
Priority:	High
Pre Condition:	None
Input:	N/A
Operations:	N/A
Expected Output:	N/A

Name:	6.3.2 OS Compatibility
Description:	The System must be compatible with the main operating system currently used in the suite, which is Windows XP. It would also be useful for the system to work with future change to Windows 7.
Priority:	High
Pre Condition:	None
Input:	N/A
Operations:	N/A
Expected Output:	N/A

Name:	6.3.3 OS Portability
Description:	There are a minority of users which use the current system using a Mac as opposed to a Windows based system. Therefore it would be useful to make the system portable to Mac OS X.
Priority:	Low
Pre Condition:	None
Input:	N/A
Operations:	N/A
Expected Output:	N/A

Name:	6.3.4 JVM
Description:	The system must be compatible with the current version of the Java Virtual Machine (Version 1.6), which is being used.
Priority:	High
Pre Condition:	None
Input:	N/A
Operations:	N/A
Expected Output:	N/A

7. Definition of terms

American Standard Code for Information Interchange (ASCII) – A numeric code for representing alphanumeric characters and symbols on a computer using single bytes.

Autoexec.bat – On modern Windows operating systems (see Microsoft Windows) autoexec.bat is a legacy method for defining important global variables, such as path names and file locations.

Batch processing – Running several programmes or processes one after another.

Command-line based applications – Software which is operated by the user typing in commands. Often this is done through MSDOS in Windows.

Durham Software for Wind Tunnels – Custom command-line based software that has been developed by the Department of Engineering for use on analysing data output from numerous wind tunnels.

Firefox – A free and very popular web browser developed by the Mozilla Corporation.

Frontend – A synonym for graphical user interface (see below) implying that the interface is the 'front' and the logic behind it is the 'back' (backend).

Functional Requirement – Particular functionality the system must provide.

Graphical User Interface (GUI) – An interface of the programme that uses text, images and menus as opposed to being purely text based. Users may use a mouse and a keyboard to interact with it.

Gigabyte (GB) – The measurement of 1000 MB (see below) in physical hardware storage on the Hard Drive.

Hard Drive – A secondary storage device used by computers in order to store data.

Hardware Platform – A particular type of hardware that the system will run on.

Input files – Configuration files with a .in extension that are read by Durham Software for Wind Tunnels.

Input Configuration File – The file used to create the GUI for a given program, by defining the input variables, which the user needs to enter.

Internet Explorer (IE) – A web browser developed by Microsoft for use on Windows machines.

Java – A programming language, which is compiled to a basic binary form called byte code and is then executed by an intervening application called a virtual machine. The virtual machine provides a level of abstraction from the operating system in use, and as a result, Java applications tend to be extremely portable between platforms.

Java Runtime Environment (JRE) – The virtual machine in charge of executing Java applications, as mentioned above (see Java).

Log File – A file which contains information about the use of the system, holds useful information such as errors that occurred.

Mac – A brand name for 'Macintosh' computers manufactured by Apple Inc. (see below Mac OS X, the operating system traditionally provided on Mac computers).

Mac OS X – A UNIX-based computer operating system with a graphical user interface, typically bundled with hardware manufactured by Apple Inc.

Megabyte (MB) – A megabyte is a quantifier for the space on a hard drive.

Memory – Memory relates to the amount of primary storage that the system has, which is the storage used by programs to load data.

Microsoft Excel – Excel is a spreadsheet application, which allows the manipulation of formulae and the creation of graphs and charts.

Microsoft Windows – A computer operating system with a graphical user interface. Notable versions include 'XP', 'Vista' and '7'.

Non-Functional Requirement – Particular performance criteria the system must satisfy.

Notepad – An extremely simple plain text editor provided with Microsoft Windows (see above).

Output files – Data output from the Durham Software for Wind Tunnels. Data can be exported into a variety of third party programmes.

Roaming profile – On networked computer systems with authentication, a user's data and personal settings are synchronized between a file server and the individual computer the user is logged onto. Hence their profile (data and settings) can 'roam' in the sense it can move between any individual computers provided it is connected to the network and can access the file server.

Software Platform – A particular type of software that the system will run on (e.g. JRE).

Web Browser – A piece of software that allows the user to view content on the Internet.

Wind tunnel – A device in which wind is blown out or through a tunnel and measurements are taken at set points. They are often used to test airflow on any number of items.

Workflow – A user-friendly term for a file describing a batch process (see Batch Processing).

W:\ drive – A shared network drive used by the Engineering Department to store wind tunnel applications and configuration files.

8. References

All Sections

Ian Summerville (2004), 'Software Engineering 7', ISBN: 0321210263.

DSW (Durham Software for Wind Tunnels) User Manual.

Project Brief.

Notes from talks with staff and students from the Engineering Department.

Domain Analysis

Rubén Prieto-Díaz (1990), Domain Analysis: An Introduction, https://users.cs.jmu.edu/prietorx/Public/publications/DomainAnalysisIntro-SEN-90.doc, last accessed 15/11/2010.

Competitors Software

Aerotech – http://www.aerotech-ate.com, last accessed 06/11/2010.

Turbulent Flow – http://www.turbulentflow.acom.au, last accessed 06/11/2010.

Hardware and Software Platforms

Webopedia (2010), 'Java', http://www.webopedia.com/TERM/J/Java.html, last accessed 18/11/2010.

Durham University ITS, 'Minimum PC Specification', http://www.dur.ac.uk/its/hardware/recommendations/minpc/, last accessed 02/12/2010.