Bournemouth University

FACULTY OF SCIENCE & TECHNOLOGY

BSc (Hons) Computing
May 2017

Synchronizing Files over a Network Using Rsync

by

Matthew Carney

Faculty of Science & Technology

Department of Computing and Informatics

Final Year Project

Abstract

File synchronization is a technology that can be adapted to software of all sizes, but no matter the scale of the product the effects can be exponential. This project applies synchronization to produce an effective solution to solve the problem of updating files across multiple computers.

The solution presented in this paper is being produced for an oil rig simulation manufacturer (Drilling Systems Ltd). The end program (NFT) uses the rsync algorithm and HTTP to propagate graphics patch files across simulation computers on a closed LAN network using a command & control architecture implemented in C#.

Also examined was the chosen methodology for this paper, Dynamic Systems Development Methodology (DSDM). Which was selected due to its ability to manage tight deadlines and changing requirements.

Both synchronization and methodology methods and literature have been examined and appropriate evaluation has been performed.

The programs unexpected complexity meant that the intended GUI could not be implemented however a strong framework for future work was complete.

Dissertation Declaration

Signed: ____.

Name: Matthew Carney (i7218006)

Date: 11th May 2017

Programme: BSc (Hons) Computing

Original Work Declaration

Signed: ____.

Acknowledgments

For Aimee, Nick, Harry and Imogen (and obviously mum and dad) you guys kept me sane.

Thank you Paul for your guidance.

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

Drilling Systems is a Bournemouth based company that creates bespoke training simulations for oil rigs. During the development of a simulator, software components are changed and updated, when the graphics program needs to be updated it must also be updated on multiple graphics computers. This task can be very time consuming, can take time away from development and can delay installation of other software on the simulator. 6 years ago a program was created to automate this task but over the years it has become outdated and lacks newer features required by the developers. The main problem will be finding a new way to transfer files effectively over a network, this will be addressed by creating a new purpose built tool which can update files on multiple computer but also provides a suit desired features.

1.1 AIM

The aim of the project is to find a solution for updating files across a network. In order to achieve this a new tool will be created that will propagate graphics patch files from one computer to the multiple graphics computers in a simulator. Additional behaviour and functionality will be derived from the client and developers who will use the program. The program will also attempt to include some form of file synchronization to improve overall transferring and operational efficiency.

The program will use a software development methodology that will guide the creation of the new program, this methodology and its effectiveness will also be evaluated.

1.2 OBJECTIVES AND SUCCESS CRITERIA

The primary objectives for this projects artefact is to produce a program that has been created around the following principles:

1.2.1 Scalability & Accessibility

The program must be built and designed in a way that the code can be easily extended and improved in the future. Crucially the code easy to read and understandable so it can be maintained by other developers.
For example, in terms of methods of communication between the programs messages must be standardized and extensible so that different version of the final program can still work together and improved in the future.

To access this objective the end program will be written using standard language practices and using external libraries to avoid ‘spaghetti code’. These standard practices can also be tested using built in tools to the many development environments.

1.2.2 Performance

The program must be resource intensive, in terms of memory and cpu usage but also in terms of network bandwidth. The program must be able to run unnoticed in the background and must not affect any simulation activity that may be occurring on the computer.

To avoid performance problems the program’s code must be well written so as to avoid any unnecessary resource usage.

This will be evaluated by using performance metrics found within the development environment a goal of less than 25% CPU usage will be expected at times of peak operational load.

1.2.3 Robustness

The program must also be able to recover from and deal with errors in an efficient and well-handled manner. Due to the nature of networked programs, errors that occur on remote programs can be hard to detect, and if they are not acted upon then they can impact and cause errors on other remote programs that rely on them.

The code in the final program should account for any possible errors that could occur within the application, should provide appropriate feed back to the user and the main program and be able to recover from the error to continue the current operation.

To test this aspect of the programming simulated errors and error procedures will be used to test the error reporting and systems in the final program.

1.3 Deliverables

- An updated transfer program
- To perform appropriate research into the tools context and use to help similar projects in the future involving transferring files over a network
- Access the effectiveness of DSDM on a project of this type

1.4 RISKS

The program being produced in this project is none mission critical piece of software; therefore delivery is not essential but any work and background research that can be done on this project will help. Due to the complexity devoting developers to this project will detract from other more pressing project that they could be undertaking, so this project aims to elevate this and provide a good ground work into what a solution may look like.

2 LITERATURE REVIEW

Due to the project having 2 points of focus, this literature review will be split into 2 areas structure to provide a context from the selected, secondary sources. The first area will consider how this project applies to literature regarding the DSDM methodology, how the literature influenced the project author's decision to use DSDM and the usefulness of the sources selected. The second area will encompass literature regarding file synchronization and how this related to the project's technical aim.

2.1 DSDM

The DSDM handbooks were published by the DSDM consortium and the editions used in this project were written by Jennifer Stapleton (Stapleton, J. 1997 & DSDM consortium 2002, 2003). The books are well written, well-presented and convey the DSDM framework well, from the basics of the framework to the more complicated intricacies of each stage. These handbooks are almost essential to understanding how DSDM works and how it should be applied. While it does not provide any critical analysis or examination of limitations of the framework it was essential to this project in providing concise information about the DSDM process.

On the other hand Sani et al (2013) addresses how DSDM considers security in its development flow and, crucially, in DSDM's own literature. In short, the paper states that 'DSDM does not support developing secure software' (pg. 1 Sani, A. et al 2013). It goes on to state that despite 'a considerable increase in security related software vulnerabilities reported over the last few years… DSDM (like other agile methods) does not provide any place to address security issues in software development' (pg. 1 Sani et al 2013). The paper also includes a comprehensive literature review that demonstrates how security seldom mentioned in publications (including publications by the DSDM consortium). To do this, Sani et al use a list of accepted security principles (Saltzer et al 1975) as a base line for their literature review, noting when they are mentioned in the publications. The review includes a set of 'Conventional Security Attacks' that can result from ignoring said principles, the networking attacks mentioned directly relate to the technical approach of this project (as networking will be involved in the final solution) and so should be considered during the design.

Paetsch et al (2003) covers different methods of requirements gathering and elicitation used in agile methodologies. DSDM is mentioned and its use of requirements prioritization (MoSCoW) and prototyping is examined. Even though information pertaining to DSDM is not very in depth and DSDM is mentioned in more of a comparative context, overall this paper did provide useful guidance for the requirements gathering and elicitation stage of the project.

Qureshi et al (2016) is another paper that compares DSDM in the context of agile methodologies in general. For each methodology the paper states the main limitations based on a series of case studies and references that Qureshi et al has found. For DSDM it is stated that DSDM ‘does not handle the engineering of average and complex project.’ (pg. 2 Qureshi et al 2016) the paper also mentions that there is no support for sizable teams. It is also stated that DSDM ‘has shown effectiveness for developing business applications’ but ‘is ineffective to develop scientific and engineering applications’ (pg. 3 Qureshi et al 2016). Generally the paper would be more relevant if these observation were explored in more depth but the points raised about DSDM are very intriguing.

Plonka et al (2014) discusses using DSDM in an industrial application, this fits well with this project as it provides insights into how DSDM is used in an actual software development environment and gives good contrast to this project in terms of scale and resources. The paper sets out a User Experience (UX) project undertaken by a software development company that conducts projects using the DSDM methodology. The project covered in the case study is being conducted to access DSDM’s effectiveness at handling UX type projects. It should be noted that the variation of DSDM used here is DSDM Altern, which works in essentially the same way. The project presented is said to benefit from DSDM’s iterative roles and its ability to adapt to changing requirements involved in UX design.

The project present by Plonka et al uses all the standard DSDM development phases but with more time and resources spent on each stages such as multiple interview stages, multiple teams and roles etc. The study highlights the use of prototypes for UI mock-ups and notes that prioritization did work at ensuring delivery of features (with some delays). An interesting point made by Plonka et al was how, even with the techniques used by DSDM to handle changing requirements, clients did not always feel they got what they asked for. Another observation made was that constant communication is emphasised in DSDM, but just because constant communication exists does not mean it is useful. The paper also mentions the effects of DSDM’s over analysis of requirements, which doesn’t mitigate problems with requirements that only occur during implementation.

2.2 File Synchronization

Purnama et al (2016) outlines a real world application of synchronization, the paper demonstrates Moodle’s (a popular Learning Management System) use of Rsync and Rdiff algorithms to speed up components of their network. The paper state that Moodle’s internal network was being strained by full transfers of course backups and a solution was proposed to use RDiffDir to synchronize their backup archives and Rsync to handle general synchronization of file systems between internal servers. The problem present here is very similar to the problem presented in this paper, seeking to improve performance of a transfer by using synchronization instead of full file transfers. Purnama et al demonstrated that using the 2 algorithms lessened the load on Moodle’s network and reduced data that had to be sent when backups were being propagated. The paper also shows the use of RDiffDir to handle synchronization of compressed archives, this is relevant for this project as one of the key file types that will have to be sent will be Unity compressed asset files so RDiffDir could be used in the final solution.

On the other hand Asubramania et al (1998) puts file synchronization in a narrower focus, putting emphasis on the use of file synchronization in propagating updates among mobile devices. This is useful as it gives a clear look at a more complex and developed synchronizer model and shows what another real world system would look like. Asubramania et al provide, in detail, different types of synchronizers and give comprehensive coverage on components, considerations and synchronization theory. The paper, though technical, is too specific in scope to have much use outside of its unique application.

Tridgell et al (1996) outlines the Rsync algorithm, the paper clearly explains what problem it is designed to address, how it works and the theory behind the algorithm. It continues by explaining step by step how the algorithm actually works which is extremely useful to comprehend when writing code involving Rsync. The clarity of the paper is one of its main merits, the way it breaks down a very complex process into easy to understand and simple to comprehend ideas is extremely admirable. It is not an easy subject and papers on file synchronization can be seen to struggle to convey their points as effortlessly as this paper. It gave a great insight into Rsync that will not only be useful for designing and implementing but also for explaining to clients how the process is going to perform.

Khanna et al (2007) present a paper that presents the DIFF3 algorithm and analyzes its behavior. This paper starts by explaining Diff’s context within the realm of synchronization tools and its general use, while the rest of the paper is an in-depth, detailed look at Diff3’s behavior and an examination of its properties. Ultimately at this point the papers usefulness in the context of this project drops off. This paper gives an excellent analysis of how another synchronization algorithm, besides Rsync, works and behaves. Khanna et al do give a very technical look at the Diff3’s mannerisms but this information is too technical and slightly out of scope for the type of synchronizer needed for this project.

3 METHODOLOGY

This chapter will outline the process for selecting an appropriate, possible methodologies and justification for the final selection.

3.1 POSSIBLE METHODOLOGIES

Software methodologies are detailed processes designed to ensure software is delivered on time and to a good standard, each methodology has its own emphasis on a particular stage of the software development lifecycle. Choosing a methodology is one of the most important aspects of any development project, as the choice of methodology will determine the flow of the project, its design, development.

3.1.1 Scrum

Scrum is an agile methodology that is designed to provide a framework for managing one or many self-contained teams. Teams consist of around 7 members each with a unique role, each team manages their own time and aims to achieve an iterative goal in a fixed period called a sprint (normally around 30 days long) (Scrum.org 2017). Scrum was not used for this project as it really is designed for teams of multiple people in order to work effectively and as this project only has one developer scrum is not an appropriate method.

3.1.2 DSDM

Dynamic Systems Management Method (DSDM) is an agile software methodology designed with continuous user involvement in mind to ensure that software is delivered on time, in budget and to user requirements usually within a business environment. The methodology was originally created to provide a public domain framework for the Rapid Application Development (RAD) methodology that was popular in the early 90s (Jennifer Stapleton 1997). DSDM attempts to continuously adapt to user requirements and uses prioritization and development cycles to ensure that the most essential features are guaranteed to be implemented.

For this project DSDM was selected for this project due to its ability to ensure software delivery even on tight deadlines, its continuous adaption to changing requirements and its ability to ensure the most critical requirements are fulfilled (DSDM Consortium 2003) are crucial to this project.

3.2 JUSTIFICATION

In the official DSDM version manual (DSDM Consortium 2002), a list of project characteristics is provided where DSDM is the correct or effective method to use for a project. For each characteristic (interactive, clear user group, decomposable complexity, compartmentalization, time constrained, prioritization, and changing requirements) listed in the handbook, the following list looks at what they mean and accesses how they apply or do not apply to this project:

1. **'Interactive, where the functionality is clearly demonstrable at the user interface'**
   - DSDM is particularly effective when it comes to projects that are UI focused. This is because changes to the UI are clearly demonstrable to the client and can aid in user involvement in the project. In DSDM functionality should be clearly demonstrable between prototypes so clients can give stronger feedback on prototypes.
   - In complex back end systems DSDM may be less of an effective method as demonstrating changes to back end code can be harder for users to understand changes and provide valuable feedback.
   - Projects where functionality is clear are good contenders for DSDM

   Suitability for project:
   - *This project will consist of a front-end UI and backend system for transferring to support it*
   - *The UI code will benefit from the use of DSDM as functionality of the UI will be clearly demonstrable and use of DSDMs development cycles will be effective for constructing a UI that the user wants. Therefore, demonstrating use of DSDMs core principles of user involvement and clear iteration throughout development*
   - *Even the backend code of the project will have a large effect on the overall functionality which should be easy for the user to see the changes to the transfer mechanics could result in time differences the general flow of the program.*
   - *DSDM Characteristic Suitability:* ***PASS***

2. **'Has a clearly defined user group'**
   - The main concern being addressed here is DSDMs need for a clearly defined and accessible user base, because DSDM is so user focused the danger of gaining the wrong viewpoints and missing important aspects of the project could violate DSDMs 4th principle and cause the final product to miss the actual needs of the users.

   Suitability for project:
   - *The original tool for this projects problem has a very clear and known group of graphics coders (the ones who apply the patches to the simulators, these programmers use the*

program daily on site). These are contactable and known to the developer of this project and they also have a clear understanding of the tool and know how they want the program improved.

- *DSDM Characteristic Suitability:* ***PASS***

3. **'If computationally complex, the complexity can be decomposed or isolated'**

- DSDM is designed to be used on systems of varying complexity, however DSDM is only suitable if the complexity of the system can be broken down into smaller, more isolated segments to reduce the overall complexity of the system.

Suitability for project:

- *The project is not the most complex but will involve multiple different systems working in tandem so as to transfer files in an effective and timely manner.*
- *Due to the nature of the C# as an object orientated language, the code will be split up into the appropriate areas as the program is written. Backend networking code will be split into its own classes separate from the threads and code of the UI and other overhead components of the program so overall complexity of the program can be reduced and used within DSDM*
- *DSDM Characteristic Suitability:* ***PASS***

4. **'If large, processes the capability of being split into smaller functional components'**

- DSDM's focus on clearly deliverable functional prototypes along with teams that are meant to work in parallel on parts of the project mean that larger project must have the capability to be split up into smaller more manageable chunks (this relates to the previous characteristic).

Suitability for project:

- *As the project is not particularly large and the team is not very extensive, this characteristic does not really apply to my project but like in the characteristic 3, the projects complexity can be broken down into clearly demonstrable chunks so even if it doesn't apply the project is still suitable for generally DSDM*
- *DSDM Characteristic Suitability:* ***N/A***

5. **'Time-Constrained'**
   - DSDM works best with projects that have fixed deadline dates, without these it is possible for schedules to slip and the fundamental benefits that DSDM provides to be lost

   Suitability for project:
   - *The project has a tight time constraints on it, not only does it need to be completed and delivered to the user before May 12th when the deadlines for the project is but it will likely have to be completed and implemented at an agreed upon date with the client. Therefore the project will clearly time constrained and will benefit from DSDM*
   - *DSDM Characteristic Suitability:* ***PASS***

6. **'The requirements can be prioritised'**
   - DSDM works best if the requirements of the project can be organised using MoSCoW prioritisation into requirements that Must, Should, Could and Will not be required for the product

   Suitability for project:
   - *This project will consist of requirements that are more important or pressing than others as using this program for many years has meant that multiple functions of varying importance have been derived by the users. There for a system such as MoSCoW will be greatly beneficial in this type of project.*
   - *DSDM Characteristic Suitability:* ***PASS***

7. **'The requirements are unclear or subject to frequent change**
   - At its core DSDM is designed to deal with requirements as they adapt during the development of software. The prevailing idea that DSDM takes from its predecessor RAD is that requirements are going to inevitably change through the development lifecycle so the framework is designed to work with these changing or unknown requirements

   Suitability for project:
   - *This characteristic is a harder to predict with this project, from the start the main requirements of the software were clearly defined. However from experience with working with the client it is known that their products do not always end up fitting their original intentions and their requirements are constantly changing on their own line of products, so it is a fair assumption to make that requirements for this project will change over the course of development.*
   - *DSDM Characteristic Suitability:* ***PASS***

4 REQUIREMENTS AND ANALYSIS

This chapter will be an overview of how the requirements for this project were gathered, what requirements were determined how they were prioritized. Existing solutions for the problem and possible technologies for the final solution will be examined.

4.1 CUSTOMER REQUIREMENTS

Meetings with the client took place twice at the Drilling Systems offices in Bournemouth, one pre-project feasibility meeting and another mid project meeting. Email contact was maintained throughout the project to answer any queries. A design document with general requirements had been produced by the graphics developers before the project, so requirements categorisation and prioritization was done in the initial meeting with the client.

4.1.1 Functional and Non-Functional Requirements

From the aforementioned design document the following core requirements were confirmed:

Functional:

1. Transfer one computer to multiple remote computers over a closed LAN network
2. Transfer multiple files of varying sizes
3. Transfer files based on blacklist rule sets
4. Automatically find computers to transfer to
5. Has specified GUI

Non-Functional:

1. Backup and rollback functionality

4.1.2 MoSCoW Requirements

MoSCoW is a means of prioritizing requirements into requirements that ‘Must, ‘Should, ‘Could’ or ‘Won’t’ be implemented. MoSCoW was used here as means of ensuring that the most crucial features were identified and prioritized during implementation due to projects tight deadline. The following requirements were derived from the previous design document and prioritized during the initial meeting.

Prioritization	Requirement	Notes
Must Have	The ability to transfer files to multiple computers on a LAN network	
	Be able to scan network to find computers to transfer to	Only certain computers should be transferred to, the ability to scan and find these computers is a critical function of the program
	Blacklist file compatibility	Certain graphics files do not need to be updated every time, so the new tool should have the ability to ignore certain files or file types
	Can work with existing setups	The program should be able to work with existing graphics installation (Eg on older simulators)
	Can handle transferring Unity and Tempest sized patch files	Should be able to transfer the types of files commonly used in Drilling Systems graphics programs
	A GUI	
	Log activity	Added during implementation
Should Have	Some form of error reporting from client programs	
	File overwriting	
	Log window to info runnings of program	
	Remain stable through any problems that could occur on the network	The program should be error prone and should be able to recover from network error
Could Have	Backup functionality and ability to roll back to previous state	Program could have the ability to roll back to previous installation
	Validation checks of installation	Checks could be implemented to ensure that the graphics installation is installed correctly
	File comparison (only update changed files)	
	Inform user if pc is out of date	Could inform the user when the files on a computer are outdated or require updating
	Persistent list of computers to transfer to on the network	

Won't Have	One way file transfer	Won't be able to transfer with no counterpart program installed on the destination computer like the previous tool

4.2 Nature of Problem

As explained in section 1.2, Drilling Systems is a Bournemouth based company that produces multi computer simulators used to train personnel to safely work on oil rigs.

When graphics software is being developed for a simulator, the software must be updated on every graphics computer on a simulator to ensure a consistency. Some simulators can consist of up to 20 graphics computers powering huge multi display setups called ‘V-walls’.

The problem is that updating files on every graphics computer can be a lengthy and time consuming process when performed manually, so a tool was developed in house to transfer files to all the necessary computers on a simulator from one computer.

4.2.1 Existing Program

This tool is called Network File Transfer Tool, it is a console based application written in C++. It uses Windows HomeGroup, a feature which allows file sharing between computers on a network (Microsoft Support 2017). The tool also supported blacklisting of certain files that shouldn’t be updated.

After years of use, graphics developers sought additional features (section **4.1**) so a replacement had been desired for some time. There have been several attempts to create a new tool over the years but the complexity of the program demanded too much of any developer’s time and with the original developer having left the company and even the source code lost, no attempts had been implemented. So, the project was given to this papers author to examine and initiate a possible solution.

4.2.2 Network Configuration

As mentioned, Drilling System simulators consist of multiple computers each performing different roles in the simulation (sound, graphics, etc), these computers are all connected on a closed Local Area Network (LAN), so the new tool would have to work on this kind of network.

4.2.3 Files & Sizes

The program will be expected to transfer any type of files but will be primarily used to update files for one of 2 types of graphics engines used by Drilling Systems; Tempest used on older simulators and more recently Unity graphics. According to the client the following are typical file for the 2 engines:

Graphics Engine	Files per installation	Average file size (Mb)
Tempest	~1800 files	1 - 25
Unity	~150 files	10 - 500

It should also be noted that Unity uses compressed asset achieves which leads to less files per setup.

4.2.4 Operating System

All of Drilling Systems software is designed for Windows, current simulators use Windows 7 as the operating system of choice. Therefore the tool has to be built to work in a Windows environment.

4.2.5 Pre-Installed Software

Drilling Systems simulation software and tools rely on the Microsoft .NET framework so the assumption can be made that the current version .NET will be installed on the computer. This means that use can be made of the libraries available in .NET as it will be installed on simulator computers.

4.3 TECHNOLOGIES

In this section possible technologies and other components that could be used in the final solution will be examined.

4.3.1 Graphic User Interface Framework

One of the main requirements given by the client was that the tool used a Graphics User Interface (GUI) as opposed to the console based interface that the previous tool used. As C# was the chosen language there were really 2 options when it came to implementing a GUI in a C# program; WPF or WinForms.

4.3.1.1 Winforms

WinForms is the traditional method for building Windows user interfaces, it is supported by .NET and provides language independent access for creating GUIs. WinForms acts as a wrapper for the Windows API (win32 API) (Misra, A, 2016) and has been implemented since version 2 of the .NET framework (Sells et al 2006, p. 25). WinForms is easy to use, reliable and provides a consistent 'Windows' look and feel across all modern Windows operating systems.

4.3.1.2 Windows Presentation Foundation (WPF)

WPF is a recent Microsoft platform for creating modern and dynamic visual applications for Windows. WPF is a management framework built on DirectX (Sells et al 2006, p. 29), unlike WinForms, WPF uses XAML (an XML based mark-up language) to declare the layout and positioning of elements (buttons, text fields, etc.) in a User Interface (UI) (Stellman, A. 2010, p. 764). One of the main concepts behind WPF applications is the separation of UI code from actual functional code running behind the UI.

4.3.2 File Transfer

4.3.2.1 Serialization

Serialization is the method of converting an object in programming into an array of bytes. In byte form this data can be easily copied, stored or sent over a network. This was experimented with in the early feasibility stage of the project but was found to consume to much program memory to be viable.

4.3.2.2 File Transfer Protocol (FTP) and Trivial File Transfer Protocol (TFTP)

There was consideration for using a more traditional protocol for the actual file transfer of this project such as File Transfer Protocol (FTP) to actually send the files to the remote computers. FTP is a protocol that is fairly easy to implement in C# due to its common usage, example implementations of both FTP servers and clients exist on the internet (CodeProject.com, 2017 & c-sharpcorner.com 2017). However, one of the main reasons for not using FTP contains a lot of features that are adapted for internet transfer which it was felt would ultimately make using it to transfer files over a LAN more complicated than necessary.

A similar protocol was found and thought to be useful, called Trivial File Transfer Protocol (TFTP). TFTP came about during the 70s and was designed to be a completely bare bones, ‘trivially’ simple to implement system for sending files over a network. It used UDP for its original implementation but other protocols could be used (RFC 1350, 1992). TFTP allowed files to be downloaded or uploaded to a TFTP server.

4.3.2.3 Hyper Text Transfer Protocol (HTTP)

Hyper Text Transfer Protocol (HTTP) was considered as a possible transferring method using an embedded webserver to server files to programs on the network. The programs would then use standard HTTP requests to retrieve and saves the files onto the computer. Using this method was very intriguing as using different protocols to send as files and to send communication messages would be efficient and reduce traffic between programs.

4.3.3 File Synchronization

4.3.3.1 Rsync

Rsync is a popular file synchronization tool distributed with Linux operating systems (Linuxcommand.org. 2017). It aims to efficiently synchronize files across multiple hosts even on low bandwidth connections, it achieves this using a dual checksum system to work out and send only the data that is missing from files instead of sending the whole file. However a Windows version of Rsync does not exist as it does on Linux. The internal workings of the Rsync have been published (Tridgell et al 1996) and many programs exist on windows, utilising the algorithm and process behind Rsync (Unison File Synchronizer, 2017 & DeltaCopy, 2017).

4.3.3.2 Rdiff

Rdiff is a variation of Rsync, it is implemented in Linux as a command line program that gives user full access over each of the stages of Rsync (linux.die.net, 2017). Rdiff allows users to perform one of the three Rsync procedures on a file; generate a signature, create a delta and patch a file. Each of these stages and the process and algorithm behind them is exactly the same as Rsync.

4.3.3.3 Octodiff

OctoDiff is an implementation of Rdiff in written in C# (OctopusDeploy, 2016). The program functions the same as Rdiff using the same algorithm behind Rsync to perform synchronization operations on a file (signature, delta or patch). The program is open source so the code could be used in this project, Octodiff is also multiplatform so can be run on Windows, Linux or Mac.

4.3.4 Web Server

4.3.4.1 NHttp

NHttp is an open source asynchronous HTTP server written in C# (pvginkel, 2017). The code is easy to add to a project and running the server from code is straightforward and simple. It also provides a range of features and customization (custom request and response behaviour). However it does not allow a working server directory to be specified but overall a very useful code for running a small but fast HTTP server from a C# program.

4.3.4.2 Internet Information Service (IIS)

Internet Information Service a web server created by Microsoft. It is intended for deployment of applications and supports a wide range of web protocols (including HTTP, FTP etc) (Technet. 2003). It is included in most editions of windows but is not normally enabled by default on the system. IIS can be used to run small servers for the purpose of software deployment, it is maintained and still supported by Microsoft (IIS.net. 2017).

4.3.4.3 WAMP

WAMP is a set of software for running localhost web servers compatible with running PHP and other web based services (Perschke, S. 2017). WAMP is created for Windows and supports a wide range of web services which can be used for testing or as a deployment option. It provides a server pointing to a local directory and controls for operating the server.

4.3.5 Code Repository

During a meeting with the client it was decided to use GitHub to store the code for the NFT project, this would be used as a general repository during development but also give the client the option to check on the progress of the project. A private GitHub repository was used that would only allow the developer and client to view the contents of the project.

4.4 EXISTING SOLUTIONS

Below are some solutions that exist on the Windows platform that could be theoretically used for similar problems in which synchronization could be used to propagate changes. However due to the unique features required by in this projects a more tailored experience was required and most of these solution only provide one to one as opposed to one to many synchronization required in this project.

4.4.1 Unison

Unison is a file synchronization tool written for Windows and Unix, it combines the features of several different types of synchronization and management tools (such as Subversion, CVS and Rsync) (Unison File Synchronizer, 2017). Unison allows 2 collections of files on separate devices and or operating systems to be kept identical when changes are made to either collection and is also implemented on both Windows and Unix with an operating system to show changes and perform actions.

4.4.2 DeltaCopy

DeltaCopy is an Rsync implementation for windows, it provides a wrapper around the existing Rsync program that allows it to run on windows operating system (DeltaCopy, 2017). The solution provides a wide range of features such as scheduling, email notifications along with a GUI all of which is integrated as a Windows service. DeltaCopy also provides full source code with the possibility to use the wrapped rsync program in other windows applications.

4.4.3 Gsync

Gsync is a GUI for Rsync written using GTK graphics library, it provides all the functionality of Rsync in GUI (OPByte.it. 2017). It was originally created for Linux based systems but versions for windows and mac exist (Grsync-win. 2016). Gsync for Windows also provides the Rsync command line program as well for extra flexibility.

4.4.4 Luckybackup

LuckyBackup is a backup solution written primarily for Linux (luckyBackup. 2014) but it has also been ported to windows (however development of the port stopped before all the features were added) (luckyBackup-win. 2007). It allows users to backup and synchronize file directories using Rsync, which the program is built around. It provides a wide range of snapshot and scheduling options and is provides a simple user interface.

4.5 PROPOSED SOLUTION

The proposed solution for this project is to create an updated program which will replace the old network transfer tool, this solution will be called Network File Transfer [NFT]. NFT will use a command and control architecture to send files and instructions from a main GUI driven NFT application to compact versions of NFT running on the graphics computers. As well as providing a basic file transfer and some client request functionality improvements, NFT will also implement Rsync based file synchronization to reduce file transfer times and provide a more efficient transfer experience.

5 DESIGN

With the requirements for the new tool confirmed with the client, and the background context for the tool and possible technologies researched it was time to start the design phase. The architectural, interface and overall design decisions as well as reasoning for these decisions will be addressed in this chapter.

5.1 SYSTEM DESIGN

It was confirmed in the early stages of discussions with the client that the new NFT tool would have to require a program running on each of the computers where file will be transferred to. This is in contrast to the previous tool but it was decided to move away from using Windows HomeGroup. So it was decided to go with a more traditional client/server model by having a ‘master’ program which the user would interact with which would in turn send instructions to and receive feedback from ‘slave’ applications which would perform said instructions. This would be a command and control architecture with the master controlling the slave programs.

Designing and implementing this Command and Control model correctly would be crucial to the project being complete to any level of quality or stability. So one of the first things to work out was how it should be implemented.

5.1.1 System Architecture & Communication

The communication between the programs went through several design iteration. First 2 TCP connections were going to be used for communication between slaves and the master programs, 1 would be used for sending commands and error reporting, the other for sending files. But it was decided that writing TCP transfer code which was optimized for LAN was not possible within the time frame of the project. So it was decided to use TCP for sending instructions and HTTP requests for downloading files using an embedded web server running within the master program. From this, an initial architectural design was agreed upon to use as the basis Network File Transfers (NFT) development:

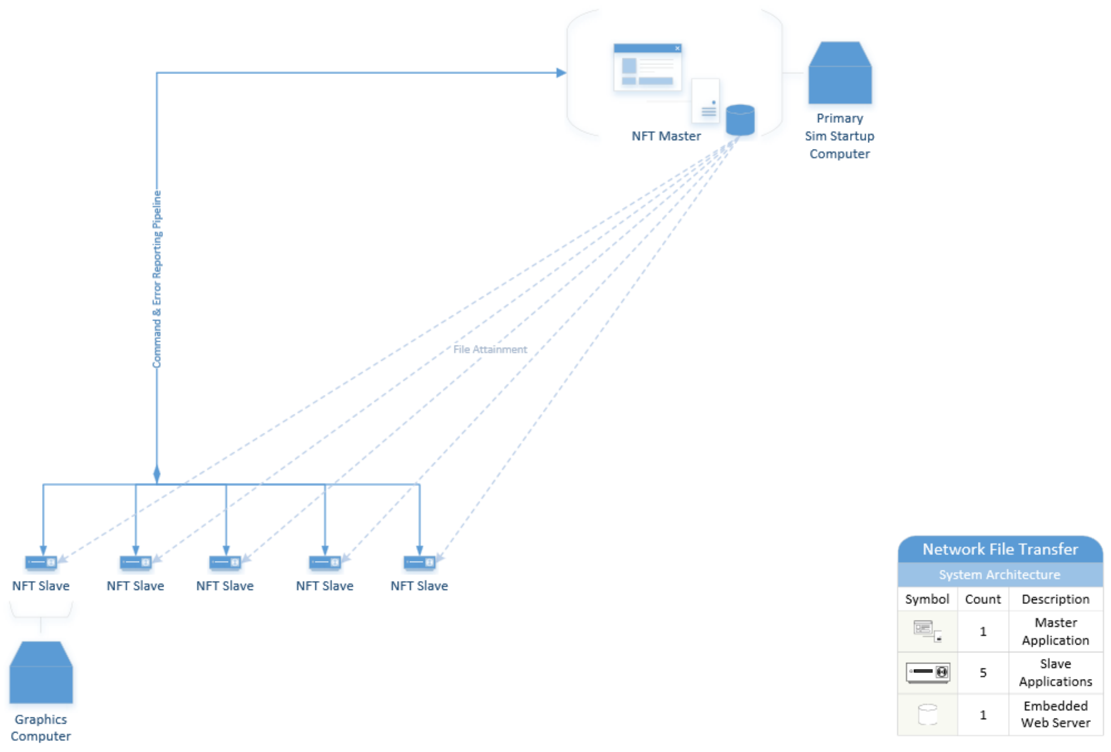


Figure 1 Initial NFT architectural layout

Next the method of communicating instructions had to be designed.

5.1.2 Command format

The command object would represent the instructions and information being sent between the programs. The object would be serialized and sent across the network where it would be de-serialized and executed by the receiving program.
This command object would contain some important attributes to carry out this task:

- Command type – Represents what type of action to be performed (transfer files, abort operation, patch files, error etc)
- File list – Stores the files that will be transferred
- Source address
- Destination address
- Message – An option field to store any additional information

5.1.3 Command receiving

For receive commands the NFT programs would use TCP listening loops that would run indefinitely to firstly listen and connect to the NFT master program (in the case of the slave program) then proceed to receive any data put on the connection stream. This data will then be de-serialized into the command object where the instruction will be processed.

5.1.4 Connection storage

The NFT master program will have to store connections to multiple NFT slave applications, in order to do this slave connection objects will have to be stored. For storage, a Slave class was designed to store the slave TCP connection medium as well as some other key information:

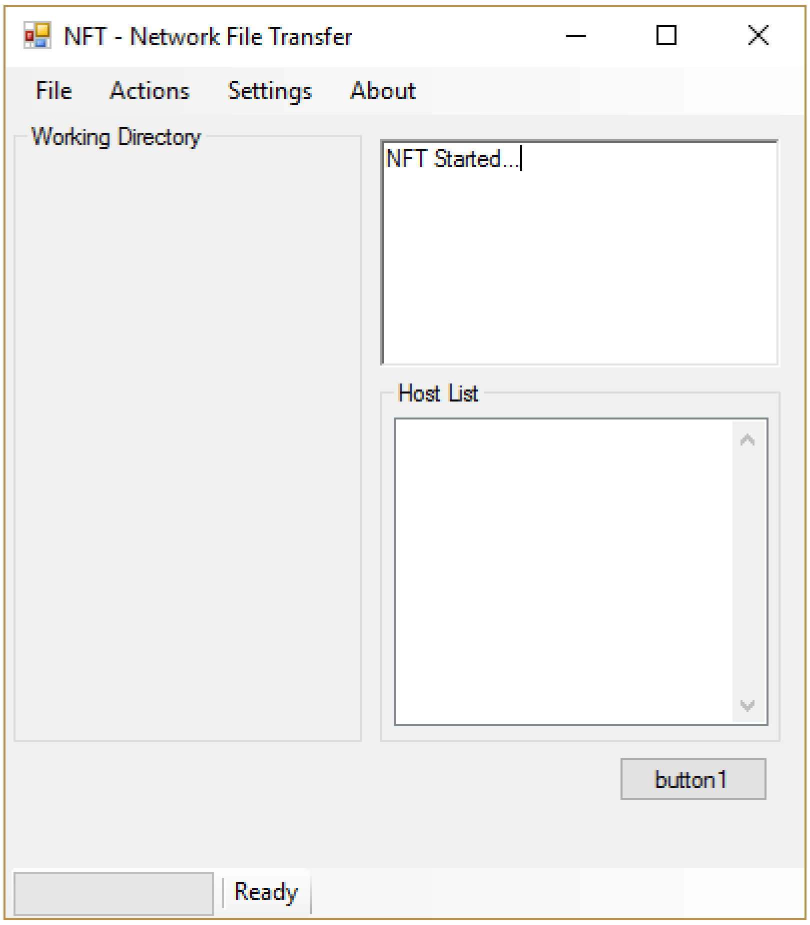
- Connection object – TCPClient or Socket object
- Network stream – Stores the stream used to send and receive data
- Slave endpoint – Stores the IP address and port of the slave application
- Connected – Represents if the slave is connected or not

This object would also store a static list that would hold all the currently connected slaves, to be used when sending commands to all slaves, displaying current connections and more functions within the program.
This connection method would only be used on the NFT master program as each slave will only be connected to one master program and not to any other NFT slaves.

5.2 GRAPHIC USER INTERFACE DESIGN

Both NFT programs will contain graphic elements in very different ways. Both will contain GUIs implemented with WinForms for functionality and consistency (see Section **4.3.1**), the master program will contain a fully-fledged GUI where the user will use the program. Whereas the slave application UI will consist a toolbar icon to provide some basic functions.

A mock program was created to experiment with what the UI may look like:



NFT Started...

Figure 2 Master program UI Mock-up

For the working directory section, a directory tree would allow the programs working directory to be set.

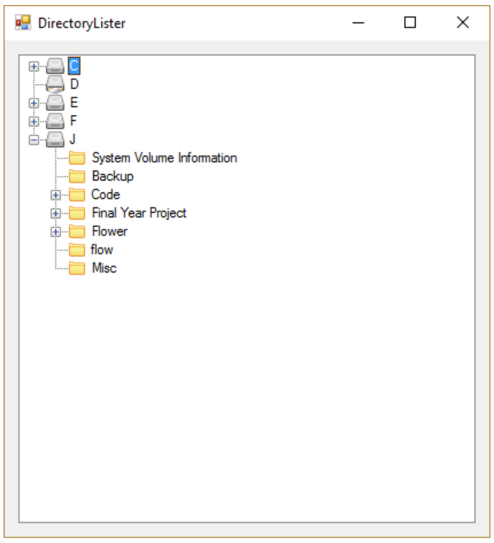


Figure 3 Working Directory view example

6 IMPLEMENTATION

6.1 OVERVIEW

The implementation of Network File Transfer (NFT) was based loosely on a client server architecture, with a main server program (NFT_Master) instructing client programs (NFT_Slave) to perform operations to their local file structure such as transfer new files, synchronize/update files and other functions. Both program used code stored in a common library (NFT_Core.dll) this contained core components of the required by both programs such as communication, file operations and the Octodiff codebase which provided Rdiff synchronization functionality.

6.2 COMMAND AND CONTROL STRUCTURE

The core of the NFT_Master and NFT_Slave program interactions are contained in the command and control structure; this section lays out how these interactions were implemented in the final program.

6.2.1 Command Layout

To implement the command messages that would be used to communicate instructions between the NFT_Master and NFT_Slave programs the class Command.cs was implemented. The following are the fields used in the final Command class:

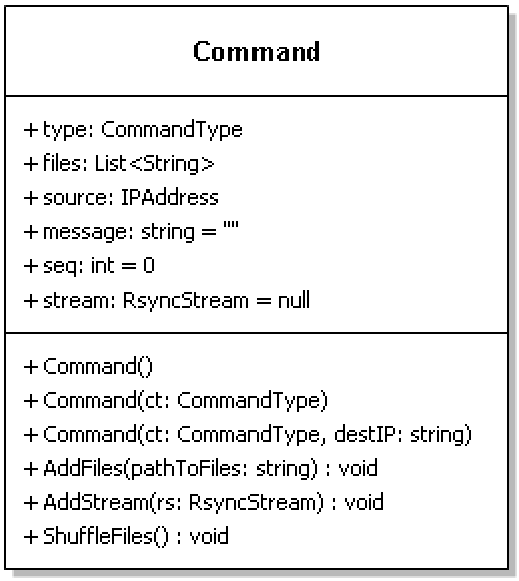

Command
+type: CommandType
+files: List<String>
+source: IPAddress
+message: string = ""
+seq: int = 0
+stream: RsyncStream = null
+Command()
+Command(ct: CommandType)
+Command(ct: CommandType, destIP: string)
+AddFiles(pathToFiles: string) : void
+AddStream(rs: RsyncStream) : void
+ShuffleFiles() : void

Figure 4 Command object implementation

The class contained all the information required communicate instructions and messages between the 2 programs (as well as data to complete these tasks). The fields for the class were built from those determined in the design state as well as a few additions that were deemed necessary to add during implementation:

- seq : A sequence number so the number of commands sent to a slave could be counted
- stream : Contained any Rsync streams if the command was being used to send Rsync data

The CommandType was implemented as an enumeration for each reference within the code, the enumeration contained the following CommandTypes:

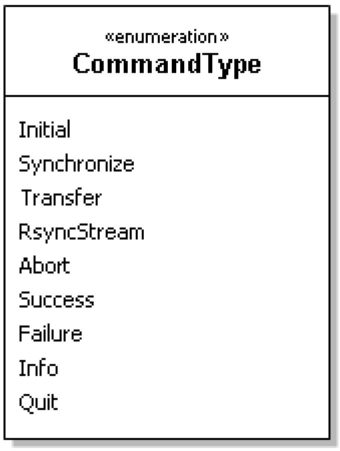
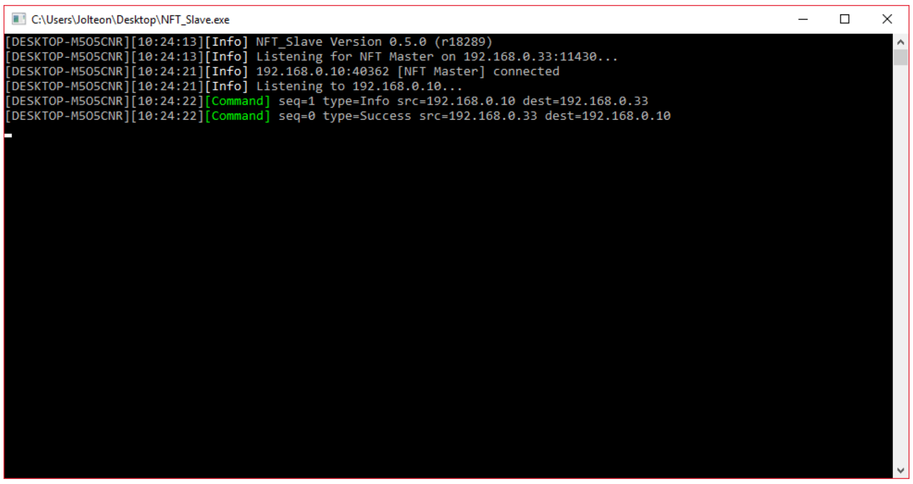


Figure 5 CommandType Enum implementation

The CommandType would be used to be determined how the command was handled and its data used once it was received.

6.2.2 Listening Mechanisms

The Command & Control connection was implemented in TCP to allow persistent connections to exist between the programs. To listen for commands, 2 listening functions were implemented; One that would listen for commands from the NFT_Master and the other that would listen for commands from the NFT_Slaves.

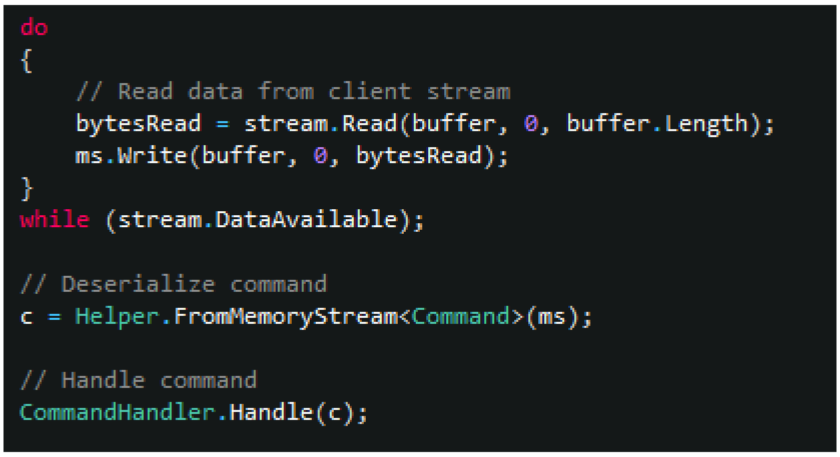


```
[DESKTOP-M5O5CNR][10:24:13][Info] NFT_Slave Version 0.5.0 (r18289)
[DESKTOP-M5O5CNR][10:24:13][Info] Listening for NFT Master on 192.168.0.33:11430...
[DESKTOP-M5O5CNR][10:24:21][Info] 192.168.0.10:40362 [NFT Master] connected
[DESKTOP-M5O5CNR][10:24:21][Info] Listening to 192.168.0.10...
[DESKTOP-M5O5CNR][10:24:22][Command] seq=1 type=Info src=192.168.0.10 dest=192.168.0.33
[DESKTOP-M5O5CNR][10:24:22][Command] seq=0 type=Success src=192.168.0.33 dest=192.168.0.10
```

Figure 6 NFT_Slave listening for and receiving commands from NFT_Master

Both listeners worked in similar ways the main difference being that the NFT_Master listener (MasterListener.cs) would have initial function that would listen for a connection attempt from the NFT_Master and would then listen for commands.

NFT_Master implemented a similar listener that would be listen for commands in a new thread once a slave had connected. Both programs implemented a similar command listening function, the function would loop infinitely receiving and handling commands. The loops would attempt to read from the stream of the connected client, this read statement would block the loop until it received data (avoiding unnessary CPU usage by the loops). The following what the core of the command listening loop looked like for both the NFT_Master and NFT_Slave listener:

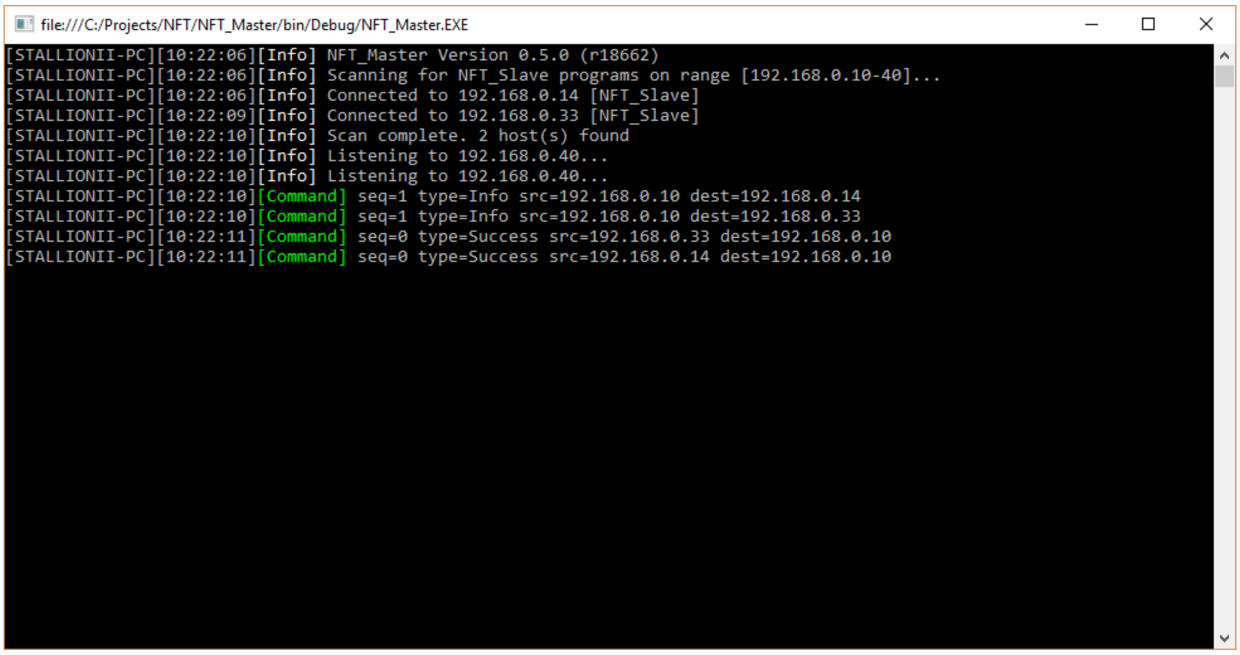


```
do
{
    // Read data from client stream
    bytesRead = stream.Read(buffer, 0, buffer.Length);
    ms.Write(buffer, 0, bytesRead);
}
while (stream.DataAvailable);

// Deserialize command
c = Helper.FromMemoryStream<Command>(ms);

// Handle command
CommandHandler.Handle(c);
```

Figure 7 Core receiving mechanism

6.2.3 Command Handling and Execution

When a command was received by either NFT_Master or NFT_Slave the command would be handled in the same manner to ensure operational consistency. Command handling was done in the CommandHandler.cs class. This class contained a method called handle which would read the command and using its CommandType determine the correct action for that command. CommandHandler.cs was also responsible for sending commands, it was deemed that this was an appropriate place to store the command sending function as to avoid duplicate code and possible alteration of command sending which would need to remain consistent for both NFT_Master and NFT_Slave.



```
file:///C:/Projects/NFT/NFT_Master/bin/Debug/NFT_Master.EXE
[STALLIONII-PC][10:22:06][Info] NFT_Master Version 0.5.0 (r18662)
[STALLIONII-PC][10:22:06][Info] Scanning for NFT_Slave programs on range [192.168.0.10-40]...
[STALLIONII-PC][10:22:06][Info] Connected to 192.168.0.14 [NFT_Slave]
[STALLIONII-PC][10:22:09][Info] Connected to 192.168.0.33 [NFT_Slave]
[STALLIONII-PC][10:22:10][Info] Scan complete. 2 host(s) found
[STALLIONII-PC][10:22:10][Info] Listening to 192.168.0.40...
[STALLIONII-PC][10:22:10][Info] Listening to 192.168.0.40...
[STALLIONII-PC][10:22:10][Command] seq=1 type=Info src=192.168.0.10 dest=192.168.0.14
[STALLIONII-PC][10:22:10][Command] seq=1 type=Info src=192.168.0.10 dest=192.168.0.33
[STALLIONII-PC][10:22:11][Command] seq=0 type=Success src=192.168.0.33 dest=192.168.0.10
[STALLIONII-PC][10:22:11][Command] seq=0 type=Success src=192.168.0.14 dest=192.168.0.10
```

Figure 8 NFT_Master scanning for and sending commands to 2 NFT_Slave clients

6.2.4 Connection storage

Both NFT_Master and NFT_Slave had to store their respective connections in a safe manner in which information could be retrieved and sent without abruptly closing the connection.

For NFT_Slave this information was stored in the MasterListener, it contained basic information about the connected slave and exposed the objects required to send and receive data from the connected NFT_Master. The following is the information stored and used by the NFT_Slave:

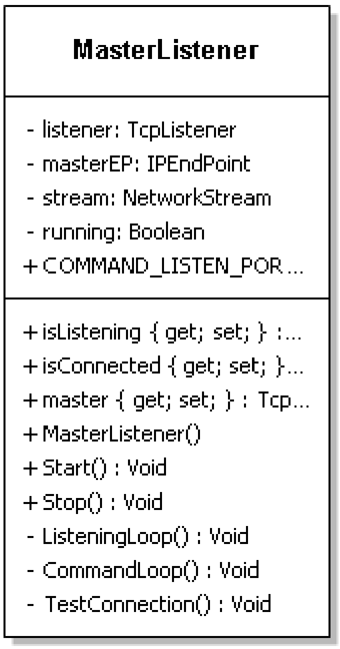


Figure 9 MasterListener implementation

NFT_Slave information was stored on NFT master in the Slave.cs class. Like the MasterListener counterpart, this contained information pertaining to the NFT_Slave and its connection medium, the class also contained several methods used for general slave operations such as connecting, disconnection, sending commands.

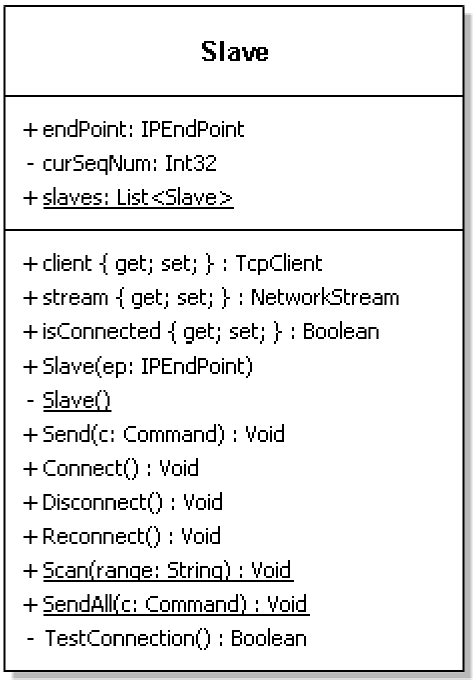

Slave
+ endPoint: IPEndPoint
- curSeqNum: Int32
+ slaves: List<Slave>
+ client { get; set; } : TcpClient
+ stream { get; set; } : NetworkStream
+ isConnected { get; set; } : Boolean
+ Slave(ep: IPEndPoint)
- Slave()
+ Send(c: Command) : Void
+ Connect() : Void
+ Disconnect() : Void
+ Reconnect() : Void
+ Scan(range: String) : Void
+ SendAll(c: Command) : Void
- TestConnection() : Boolean

Figure 10 Slave implementation

The class also contained several static variables and methods that were used to control all the connected slaves, these included; the scan method to scan for slaves on a given network range, the sendToAll method to send commands to all connected NFT_Slave programs. It also contained a static list that stored all the currently connected Slave application called 'slaves'.

6.2.5 Error Handling

After the Listeners had been implemented it was realized that using the Command & Control connection to report errors could clutter the crucial connection. Since error reporting was less crucial to the internal workings of the system but should still be shown for the users sake, it was decided to use UDP to send general error messages back to the NFT_Master program to inform the user.

Serialization would be used to send custom Error objects to the NFT_Master on a hard coded port (the source address from the Command would be used to determine the sending address). The structure of the error object is laid out in Error.cs:

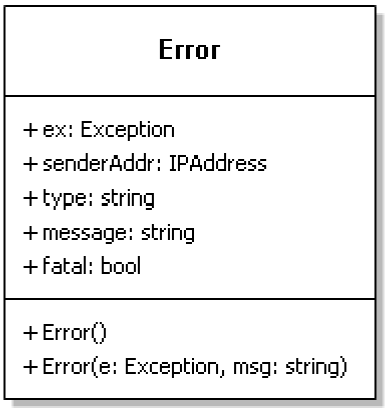
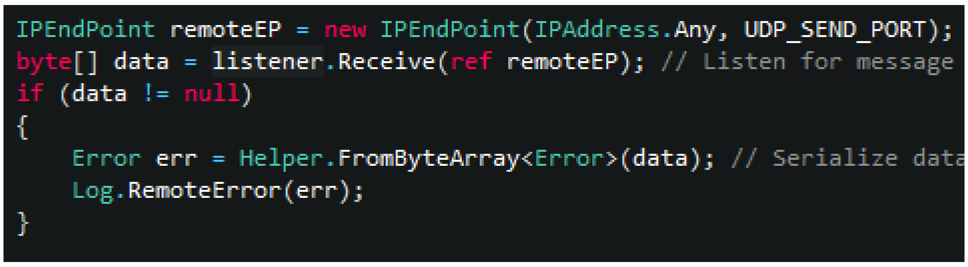


Figure 11 Error object implementation

The contained information such as:

- Ex : Which stored the exception object that had occurred for further inspection
- senderAddr: which contained the machines address where the error occurred
- type : which contained the exception type in string for display
- message : Additional message about the implication of the error
- fatal : Which would store if the error had interrupted opertations

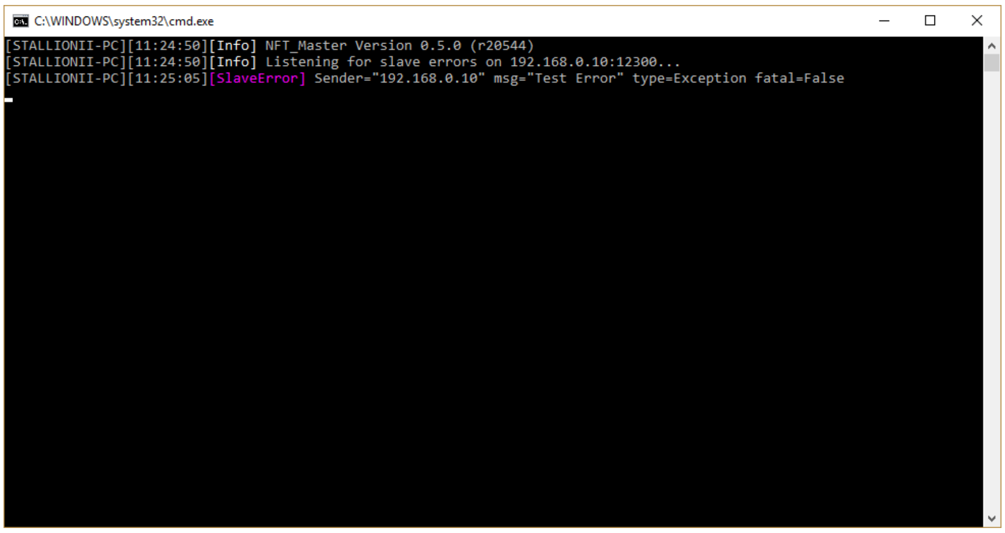
To send and receive this object ErrorReporter.cs contained 2 functions, one that would listen for incoming UDP error messages and the other that would send them. The error listener worked in a similar way to the TCP command listeners, it contained a loop that tried to receive any UDP datagrams and serialize them into the Error object



```
IPEndPoint remoteEP = new IPEndPoint(IPAddress.Any, UDP_SEND_PORT);
byte[] data = listener.Receive(ref remoteEP); // Listen for message
if (data != null)
{
    Error err = Helper.FromByteArray<Error>(data); // Serialize data
    Log.RemoteError(err);
}
```

Figure 12 UDP Error receiving logic

The sending method attempted to send the error message to the provided address, due to the nature of UDP sending did not require a connection to be established.

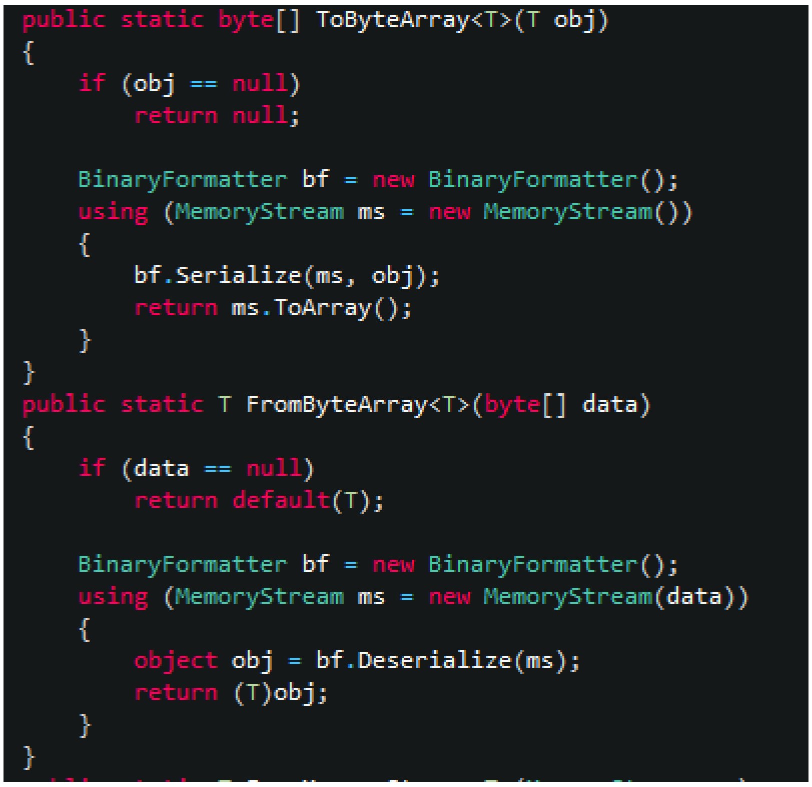


```
[STALLIONII-PC][11:24:50][Info] NFT_Master Version 0.5.0 (r20544)
[STALLIONII-PC][11:24:50][Info] Listening for slave errors on 192.168.0.10:12300...
[STALLIONII-PC][11:25:05][SlaveError] Sender="192.168.0.10" msg="Test Error" type=Exception fatal=False
```

Figure 13 NFT_Master receiving an error message from and NFT_Slave

6.2.6 Object Serialization

Due to the amount of objects that required serialization in order to be send and the different types that the objects had to serialized to, general serialization methods were implemented the Helper.cs class that could be given the type to serialize or deserialize to and then perform the necessary operations. The generic serializing functions were laid out in the following way:



```
public static byte[] ToByteArray<T>(T obj)
{
    if (obj == null)
        return null;

    BinaryFormatter bf = new BinaryFormatter();
    using (MemoryStream ms = new MemoryStream())
    {
        bf.Serialize(ms, obj);
        return ms.ToArray();
    }
}
public static T FromByteArray<T>(byte[] data)
{
    if (data == null)
        return default(T);

    BinaryFormatter bf = new BinaryFormatter();
    using (MemoryStream ms = new MemoryStream(data))
    {
        object obj = bf.Deserialize(ms);
        return (T)obj;
    }
}
```

Figure 14 Example of generic converter methods

A conversion method was also implemented from memorystreams, this was used for sending serialization data.

6.3 SYNCHRONIZATION

To send and handle serialization within the program the Octodiff code base was used. Octodiff is a console program written in C# that functions like RDiff which allows each of the stages of Rsync (signature, delta and patching of files) to be performed. In order for it to be used in NFT, Octodiff had to be modified to return MemoryStream objects instead of creating files for signatures or deltas. These MemoryStream objects could then be sent using RsyncStream object (RsyncStream.cs) that contained them and the relevant information. The fields used in this object were:

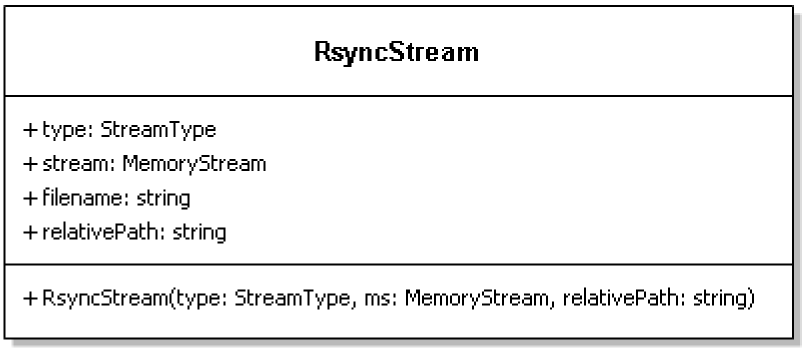


Figure 15 RsyncStream implementation

- Type : contained the type of Rsync stream (Signature or Delta stream)
- Stream : contained the stream of Rsync data itself
- Filename : the file that the stream related to or was generated for
- relativePath : stored the files location within the NFT working directory

This RsyncStream object was then stored within a command object, with the object of that command set to CommandType.RsyncStream the stream would be appropriately handled and the correct actions performed when it was received by a program.

For performing Rsync tasks the class RsyncOps.cs contained the methods for generating signatures and deltas for files and patching. This class was where Octodiff was called to perform Rsync tasks

6.4 MASTER FLOW

The NFT master program is responsible for handling user input and for instructing NFT slaves. The following were the steps the master application would use in its general program flow:

1. Slave detection
2. Constructing commands & sending
3. Listen for command messages from connected NFT_Slaves
4. Listening for errors
5. Handling GUI input

6.5 SLAVE FLOW

The general flow of the NFT_Slave program would be as follows:

1. Listen for NFT_Master program
2. Listen for commands from NFT_Master
3. Send commands to symbolize success or failure and corresponding Rsync stream data
4. Sending any errors using ErrorReporter
5. Handle incoming command

6.6 Library Layout

Due to the nature of the NFT architecture there were a lot of classes that had shared roles in both NFT applications. So all program independent classes were stored in a class library called NFT_Core.dll which would be required by both NFT programs and would store all the code and would that the code used in both classes would be the same. The library was split into the following namespaces to simply code and order sections based off the function of the code (Comms, Core, Logger, Rsync and Octodiff).

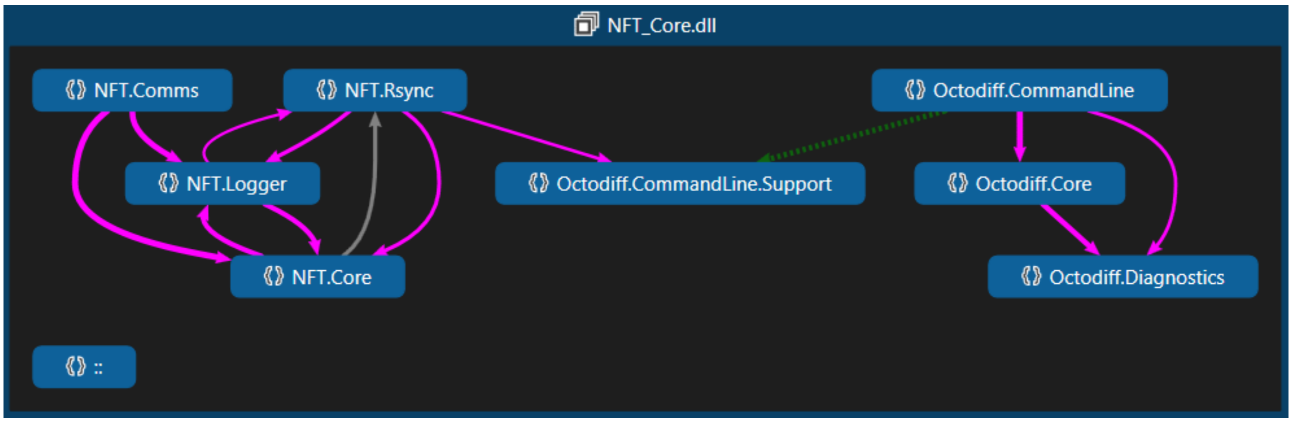


Figure 16 Code map of NFT_Core.dll

6.7 Application Settings

NFT settings were used to store options for NFTs operation this included; working directory were files would be transferred to, max number of cores to use on any multithread operation and the number of NFT_Slaves to be transferring at any time. Other settings such as port numbers were hardcoded to ensure connection consistency across the different programs and were not meant to be changed by the user.

These settings were originally going to be stored in the Windows registry, however this required administration rights so internal C# executable settings were used instead.

6.8 Issues

6.8.1 TCP connections

An issue came about when working out how to store the slave connections on the master program and how to keep these connections active during storage. At this point in development, sockets were being used to send data, when the sockets weren’t being used the connection could randomly close. Solutions to storing sockets in C# seemed too impractical, so it was decided go back to using TCPClient, this is an object in C# that acts as a wrapper for a socket while also providing more functionality. The stability of the connection with TCPClient while being stored was more consistent that using sockets so it was used the program.

Another occurred while trying to implement the listening and the command receiving logic into one loop in the SlaveListener, this was a bad idea as the loop became unnecessarily complex. It was decided to split the logic into 2 loops; one that would handle the listening and connecting to the master and the other that would receive of commands. This logic made it easier to debug and to understand the code.

6.8.2 Web Server

Multiple issues occurred while trying to incorporate an embedded web server into the NFT master program. Microsoft IIS server was selected due to its ability to point the web server at a given path, in order to configure the IIS server from C# the Microsoft.Web.Administration.dll library had to bereferenced. This library can only be found in at a specific path only when some basic IIS features have been enabled. Once this was referenced, IIS could now be configured from within the program, initially it was slightly confusing to understand how IIS was laid out. Changes made to IIS within the program would persist even after the program was closed so it was clear that the appropriate fail safes and sanity checks had to be included.

The next problem was that even with IIS server running it would refuse any connection. Also at this program also required administration rights to configure IIS without crashing, a simple procedure using app.manifest file to request that NFT_master would be invoked as an administrator. In the end IIS was put on hold the problems mentioned could not be overcome within the time frame, so WAMP was used as an external web server.

6.8.3 Synchronization

While implementing synchronization it was realized the original plan to send a signature file to all slaves and then remotely calculate the new file deltas would not be possible due to the way Rsync was designed. Signatures of files had to be sent from the NFT_Slaves and the deltas calculated on the NFT_Master and then sent back.

At first it was thought that UDP could be used to send RsyncStream objects (in a similar fashion to the ErrorReporter) but it was soon realized that RsyncStream objects could surpass the max size of a UDP packet. Also realizing that delivery of this data was crucial to the working of the system a method of sending an RsyncStream within a Command was implemented.

7 EVALUATION

7.1 DSDM EVALUATION

In this section DSDM effectiveness and relevance for this project will be examined.

7.1.1 Justification Evaluation

During the methodology selection phase a set of attributes for ideal projects for DSDM were listed from the DSDM handbook (Section **4.1.2**). These attributes were assessed against the project at that time and in this section this list will be revisited and reassessed.

1.) 'Interactive, where the functionality is clearly demonstrable at the user interface'

Original DSDM Characteristic Suitability: ***PASS***

- Like in the original evaluation of these characteristics, the GUI of the solution would clearly demonstrate functionality in the way DSDM would want. However as the GUI was not implemented during this project that does not apply.
- The backend system would not also not be very clear even when large parts of the code was changed

Evaluated DSDM Characteristic Suitability: ***FAIL***

2.) 'Has a clearly defined user group'

Original DSDM Characteristic Suitability: ***PASS***

- The user group of the original program has not changed since the beginning of this project, so the user group is still clearly definable

Evaluated DSDM Characteristic Suitability: ***PASS***

3.) 'If computationally complex, the complexity can be decomposed or isolated'

Original DSDM Characteristic Suitability: ***PASS***

- In the original analysis of the project the complexity of the solution was vastly underestimated, it was assumed that the end solution would not be that complex and it was also assumed that any complexity could be split up due to the nature of the programming language used.
- Both of these assumptions were incorrect, C# encourages code to be grouped files with other code of similar functions but this does not mean that the overall code can be split up so neatly. In order to implement any features the group work for the entire system had to be laid out.

Evaluated DSDM Characteristic Suitability: ***FAIL***

4.) 'If large, processes the capability of being split into smaller functional components'

*Original DSDM Characteristic Suitability: **N/A***

- Much like the previous characteristic, the overall complexity of the solution was underestimated, and the ability to split up any complexity was also not truly realised
- When development of the solution started it was

*Evaluated DSDM Characteristic Suitability: **FAIL***

5.) 'Time-Constrained'

*Original DSDM Characteristic Suitability: **PASS***

- The project did have a clearly defined deadline however the deadline for the code was too flexible for DSDM which prefers strong deadlines that cannot change, this forces final development cycles to be derived early and be stuck to throughout the project
- Due to the none mission critical nature of code a tight deadline was not imposed by the client and during development, cycles and their deadlines changed and were not adhered to.
- So although the project was time constrained, not all the timings in the project were as constrained as DSDM required

*Evaluated DSDM Characteristic Suitability: **FAIL***

6.) 'The requirements can be prioritised'

*Original DSDM Characteristic Suitability: **PASS***

- The requirements for the project could be very easily prioritized and this process helped greatly throughout development especially for ranking the most important requirements to be implemented

*Evaluated DSDM Characteristic Suitability: **PASS***

7.) 'The requirements are unclear or subject to frequent change

*Original DSDM Characteristic Suitability: **PASS***

- It was assumed that these requirements would change throughout the project due to the nature of development within Drilling Systems, however the user base had been using the program for many years so had a very clear idea of what needed to be added.
- These requirements did not change throughout the project, if a GUI had been completely implemented that may have changed as the client refined their ideas of what they wanted the UI to look like and do but the requirements of the underlying system where known and didn't need to change

*Evaluated DSDM Characteristic Suitability: **FAIL***

7.1.2 Development Cycles

During development, DSDM development cycles were used to ensure that requirements created. 3 cycles were planned; the first to implement file transfer, the second the implement synchronization and the third to implement a GUI. The first cycle was meant to last 2 weeks, the second and third lasting 1 week. As with DSDM a prototype was to be delivered at the end of each cycle to give to the client who would then test and give feedback.

It was soon realized in the first cycle that the functionality of the program could not be split up so neatly and that addition system architecture had to be implemented before any of the cycles aims could be added. This additional code (unrelated to the functional aim of the cycle) was crucial and had to be implemented correctly before anything else, this caused the deadline for the first cycle to be missed and the second, eventually it was decided to scrap the cycles as working on the general architecture was the only way the program could be developed

7.2 Artefact Evaluation

To evaluate the end artefact of the project, comparison of how many MoSCoW requirements were implemented in the final program and examination of how well the original solution objectives will be performed.

7.2.1 MoSCoW Requirements

Prioritization	Requirement	Implemented	Notes
Must Have	The ability to transfer files to multiple computers on a LAN network	🟧	With WAMP activing as a temporary webserver on the NFT_Master computer, file transfer is possible
	Be able to scan network to find computers to transfer to	🟩	
	Blacklist file compatibility	🟥	
	Can work with existing setups	🟩	NFT has to be set to the path of the existing installation
	Can handle transferring Unity and Tempest sized patch files	🟩	
	A GUI	🟥	
	Log activity	🟩	
Should Have	Some form of error reporting from client programs	🟩	
	File overwriting	🟩	
	Log window to info runnings of program	🟧	Log system output just needs to be piped to window
	Remain stable through any problems that could occur on the network	🟩	
Could Have	Backup functionality and ability to roll back to previous state	🟥	
	Validation checks of installation	🟧	Files will be validated against those on master computer but standalone validation was not implemented
	File comparison (only update changed files)	🟩	Handled by Rsync
	Inform user if pc is out of date	🟧	Won't inform user but Rsync will only update files that have changed
	Persistent list of transfer targets on the network	🟧	Addresses of hosts will be known when discovered so could be stored in GUI
Won't Have	One way file transfer	🟥	

7.2.2 Objectives

7.2.2.1 Scalability & Accessibility

Core code for the solution was implemented in a shared library, this code can be easily added and accessed from the NFT programs with fairly minimal effort. The coding standards and naming conventions in the program adhere to the Microsoft coding standard for C# and no warning on standard practices were raised in the final solution

7.2.2.2 Performance

As this is not the final version of the program the true performance of every aspect of this project cannot be truly. However the most resource intensive operations of the program can be tested and the results analyzed.

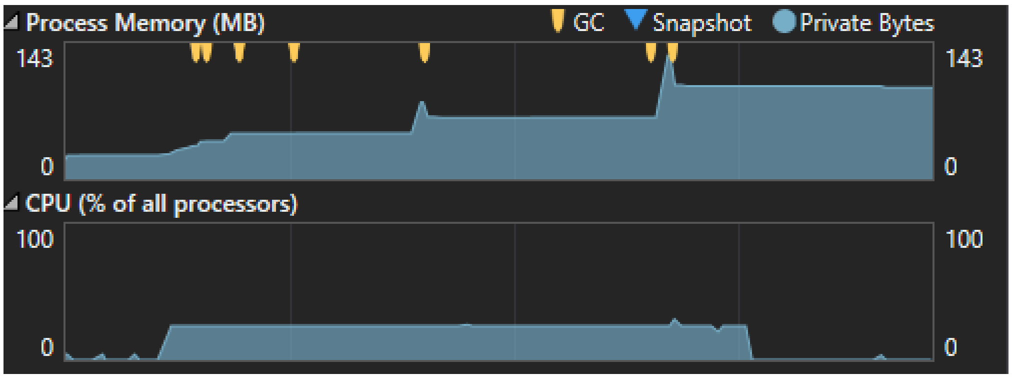


Figure 17 Resource usage during Rsync signature generation

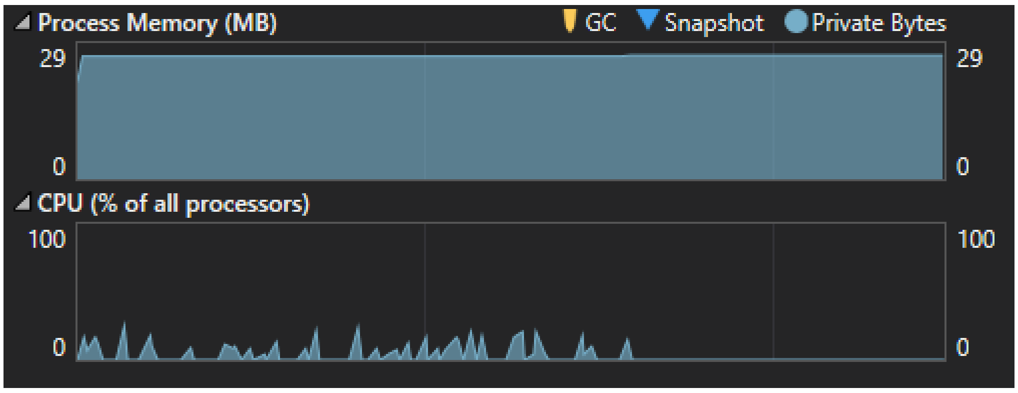


Figure 18 Resource usage during HTTP file transfer

During both file transfer and Rsync signature generation CPU remained below the goal of 25% CPU usage (although memory usage did increase during signature generation).

7.2.2.3 Robustness

An error handling and reporting system was implemented in the final solution, error handling was included in all parts of the program that critical errors could occur and mechanisms were put in place to recover from errors

7.2.3 GUI

Due to the complexity of the underlying system, the GUI was not implemented before the deadline. However due the NFT architecture that has been designed with multiple hooks and methods to allow for a GUI and other programs to be built on top of the system and use the components.

8 CONCLUSIONS

8.1 DSDM

DSDM was not the most effective methodology for this project, the project was too complex, requirements were not subject to much change and the deadlines involved as time constrained as intended. All these factors lead to DSDM not working correctly. These factors were not fully realised at the beginning of the project. DSDMs methods of requirements gathering and ranking (using MoSCoW) were very useful however and crucial for focusing on functionality during the development phase. As mentioned in (Qureshi et al 2016) DSDM does not work for complex projects such as this, this was made worse by the end solutions intertwined code that means that complexity could not be broken down. For a project such as this a looser, agile based framework would have been more appropriate.

8.2 ARTEFACT

Although the artefact did not implement every requirement defined by the user it provided a good framework and basis for a new file transfer tool. The library code provides a solid framework for a GUI to be built on top of. One the aims of this project was to implement a new tool and the program created here provides a good core for a new effective transfer tool, and can be enhanced with future work.

8.3 FUTURE WORK

8.3.1 Custom communication protocol

If more time could be afforded to the development of NFT, it would be essential to implement a custom TCP transferring code. This would allow much more controlled transferring experience and would be a good experiment in writing file transferring code in C#. The code would support multithreading, multiplexing and error recovery, writing bespoke transfer code would also allow unneeded features to be removed for better performance and would contribute general code simplicity.

8.3.2 Implement a GUI

The NFT system that has been produced in this project has been designed to allow entry points for other programs to use the code base (3rd party programs, GUIs). Static functions and object have been designed to be assessed by an overlying GUI, this is in fact crucial to access all the functionality of NFT. Now that the underlying architecture has been established a GUI can be more quickly implemented to the client's specifications. DSDM can even be used here to produce UI prototypes to determine the ideal layout for the client and end users

8.3.3 Embedded Webserver

WAMP was used as a temporary web server in the final solution but this would be replaced with a webserver which could be started and controlled from within the program.

Word Count: 10000

REFERENCES

Asubramania, S, B. Pierce, B, C. 1998. 'What is a File Synchronizer?' *Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '98),* (507), pp. [Online]. Available at: http://www.cis.upenn.edu/~bcpierce/papers/snc.ps (Accessed: 2nd March 2017).

CodeProject.com. 2017. *Simple C# FTP Class - CodeProject*. [ONLINE] Available at: https://www.codeproject.com/Tips/443588/Simple-Csharp-FTP-Class. [Accessed 27 April 2017].

C-sharpcorner.com. 2017. *Simple FTP Demo Application Using C# And .Net 2.0*. [ONLINE] Available at: http://www.c-sharpcorner.com/article/simple-ftp-demo-application-using-C-Sharp-and-net-2-0/. [Accessed 27 April 2017].

DeltaCopy. 2017. *DeltaCopy - Rsync for Windows*. [ONLINE] Available at: http://www.aboutmyip.com/AboutMyXApp/DeltaCopy.jsp. [Accessed 27 April 2017].

DSDM Consortium, 2002. *Business Driven Systems Development*. Edition. DSDM Consortium.

DSDM Consortium, 2003. *DSDM: Business Focused Development, Second Edition*. 2 Edition. Pearson Education.

Grsyn-win. 2016. *Grsync for Windows download.* [ONLINE] Available at: https://sourceforge.net/projects/grsync-win/. [Accessed 11 May 2017].

IIS.net. 2017. Home : The Official Microsoft IIS Site . [ONLINE] Available at: https://www.iis.net/. [Accessed 11 May 2017].

Khanna, S. Kunal, K. Pierce, B, C. 2007. 'A Formal Investigation of Diff3', *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pp. [Online]. Available at: http://www.cis.upenn.edu/~bcpierce/papers/diff3-short.pdf (Accessed: 2nd March 2017).

linux.die.net. 2017. *rdiff(1) - Linux man page*. [ONLINE] Available at: https://linux.die.net/man/1/rdiff. [Accessed 10 May 2017].

Linuxcommand.org. 2017. *rsync*. [ONLINE] Available at: http://linuxcommand.org/man_pages/rsync1.html. [Accessed 10 May 2017].

luckyBackup. 2014. *luckyBackup - backup and sync utility*. [ONLINE] Available at: http://luckybackup.sourceforge.net/. [Accessed 11 May 2017].

luckyBackup-win. 2007. *contrib:luckybackup-win - BMT Solutions*. [ONLINE] Available at: http://www.bmtsolutions.us/wiki/doku.php?id=contrib:luckybackup-win. [Accessed 11 May 2017].

Microsoft Support. 2017. *Homegroup from start to end.* [ONLINE] Available at: https://support.microsoft.com/en-gb/help/17145/windows-homegroup-from-start-to-finish. [Accessed 08 May 2017].

Misra, A. 2016. Use of Windows Presentation Foundation and Windows Forms in Windows Application Programming. *International Journal of Advanced Research in Computer Science*, [Online]. Volume 7, No 7, 1. Available at: http://search.proquest.com/openview/f67a5535ff1ef7af53e9dd2e176606f2/1?pq-origsite=gscholar&cbl=1606379 [Accessed 27 April 2017].

OctopusDeploy. 2016. *GitHub - OctopusDeploy/Octodiff: 100% C# implementation of remote delta compression based on the rsync algorithm*. [ONLINE] Available at: https://github.com/OctopusDeploy/Octodiff. [Accessed 27 April 2017].

OPByte.it. 2017. *OPByte: Grsync rsync GUI interface frontend for Linux, Windows and Mac OS X.* [ONLINE] Available at: http://www.opbyte.it/grsync/. [Accessed 11 May 2017].

Paetsch, F. Eberlein, A. Maurer, F. 2003. 'Requirements Engineering and Agile Software Development', *Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies,* 1(ISBN 0-7695-1963-6), pp. pg308-314 [Online]. Available at: http://ase.cpsc.ucalgary.ca/uploads/Publications/PaetschEberleinMaurer.pdf (Accessed: 8th February 2017).

Perschke, S. 2017. *WampServer delivers a smart, Windows-friendly platform for Apache, MySQL and PHP-based apps | Network World*. [ONLINE] Available at: http://www.networkworld.com/article/2187564/software/wampserver-delivers-a-smart--windows-friendly-platform-for-apache--mysql-and-php-based-apps.html. [Accessed 29 April 2017].

Plonka, L. Sharp, H. Gregory, P. Taylor, K. 2014. ‘Ux design in agile: a DSDM case study’. *Agile Processes in Software Engineering and Extreme Programming: 15th International Conference*, XP 2014. Avaliable at: http://oro.open.ac.uk/40418/1/XP2014CameraReady.pdf

Purnama, F. Usagawa, T. Ijtihadie, R, M. Linawati. 2016. 'Rsync and Rdiff Implementation on Moodle’s Backup and Restore Feature for Course Synchronization over The Network', *2016 IEEE Region 10 Symposium,* (), pp. [Online]. Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7519372 (Accessed: 2th March 2017).

pvginkel. 2017. *GitHub - pvginkel/NHttp: Simple asynchronous .NET HTTP server*. [ONLINE] Available at: https://github.com/pvginkel/NHttp. [Accessed 27 April 2017].

Qureshi, M, R, J. Bajaber, F. 2016. 'Comparison of Agile Process Models to Conclude the Effectiveness for Industrial Software Projects', *Sci.Int.(Lahore),* 28(6), pp. 5119-5123 [Online]. Available at: http://www.sci-int.com/pdf/20937959451%20a%201%205115-5119%20M.%20Rizwan%20Jameel%20Qureshi--COMp--KSA.pdf (Accessed: 2nd March 2017).

RFC 1350. 1992. *RFC 1350 - The TFTP Protocol (Revision 2)*. [ONLINE] Available at: https://tools.ietf.org/html/rfc1350. [Accessed 27 April 2017].

Saltzer, J, H. Schroeder, M, D. 1975. ‘The Protection of Information in Computer Systems’, 1278-1308. *In the Proceedings of the IEEE*. Available at http://www.cs.virginia.edu/~evans/cs551/saltzer/

Sani, A. Firdaus, A. Jeong, S, R. Ghani, I. 2013. 'A Review of Software Development Security Engineering using Dynamic Systems Method (DSDM)', *International Journal of Computer Applications,* 69(25), pp. [Online]. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.403.8971&rep=rep1&type=pdf (Accessed: 10th February 2017).

Scrum.org. 2017. *What is Scrum?.* [ONLINE] Available at: https://www.scrum.org/resources/what-is-scrum. [Accessed 08 May 2017].

Sells, C. Weinhardt, M. 2006. *Windows Forms 2.0 Programming*. 2nd ed. [ONLINE] Available at: https://books.google.co.uk/books?hl=en&lr=&id=Woa6WJyyvqIC&oi=fnd&pg=PT31&dq=winforms&ots=M6PczRFcdA&sig=F1XFTn64kaQSKWXTsvRS5mXdCTM#v=onepage&q&f=false. Addison Wesley.

Stapleton, J. 1997. *DSDM: Dynamic Systems Development Method: The Method in Practice*. 1 Edition. Addison-Wesley Professional.

Stellman, A. 2010. *Head First C#, 2E: A Learner's Guide to Real-World Programming with Visual C# and .NET (Head First Guides)*. 2 Edition. O'Reilly Media.

Technet. 2003. *Running IIS 6.0 as an Application Server*. [ONLINE] Available at: https://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/ddf1d92f-3e6e-423f-b024-35cefc10a22f.mspx. [Accessed 08 May 2017].

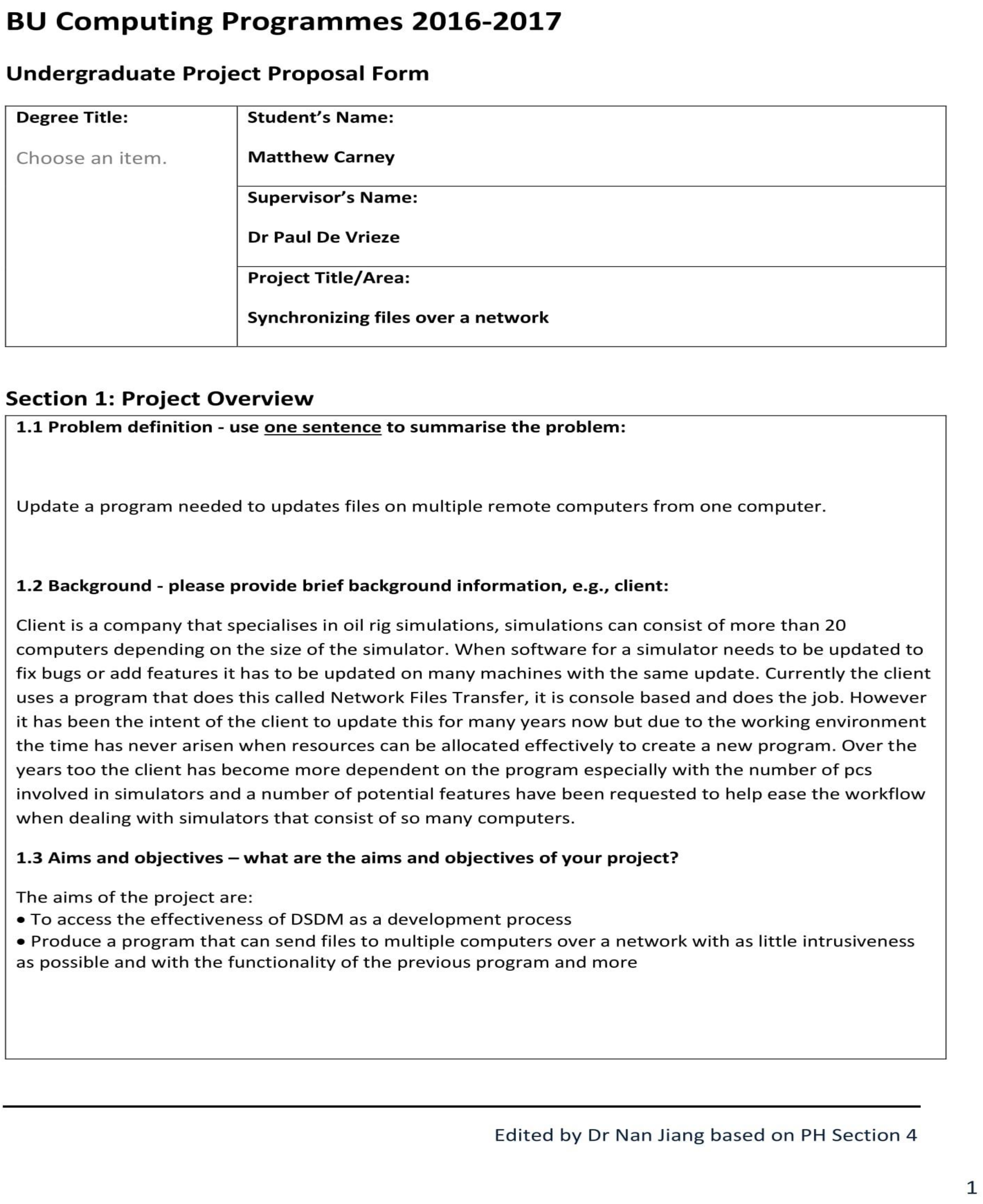
Tridgell, A. Mackerras, P. 1996. 'The rsync algorithm', *ANU Research Publications*, pp. [Online]. Available at: https://openresearch-repository.anu.edu.au/bitstream/1885/40765/3/TR-CS-96-05.pdf (Accessed: 12th February 2017).

Unison File Synchronizer. 2017. *Unison File Synchronizer*. [ONLINE] Available at: https://www.cis.upenn.edu/~bcpierce/unison/. [Accessed 27 April 2017].

APPENDIX A - CD CONTENTS

- Dissertation (PDF)
- Dissertation (Word)
- Early Concept Prototypes
- Network File Transfer Project
- Network File Transfer Builds

APPENDIX B – PROJECT PROPOSAL



BU Computing Programmes 2016-2017

Undergraduate Project Proposal Form

Degree Title: Choose an item.	**Student's Name:** **Matthew Carney**
	Supervisor's Name: **Dr Paul De Vrieze**
	Project Title/Area: **Synchronizing files over a network**

Section 1: Project Overview

1.1 Problem definition - use one sentence to summarise the problem:

Update a program needed to updates files on multiple remote computers from one computer.

1.2 Background - please provide brief background information, e.g., client:

Client is a company that specialises in oil rig simulations, simulations can consist of more than 20 computers depending on the size of the simulator. When software for a simulator needs to be updated to fix bugs or add features it has to be updated on many machines with the same update. Currently the client uses a program that does this called Network Files Transfer, it is console based and does the job. However it has been the intent of the client to update this for many years now but due to the working environment the time has never arisen when resources can be allocated effectively to create a new program. Over the years too the client has become more dependent on the program especially with the number of pcs involved in simulators and a number of potential features have been requested to help ease the workflow when dealing with simulators that consist of so many computers.

1.3 Aims and objectives – what are the aims and objectives of your project?

The aims of the project are:
- To access the effectiveness of DSDM as a development process
- Produce a program that can send files to multiple computers over a network with as little intrusiveness as possible and with the functionality of the previous program and more

Section 2: Artefact

2.1: What is the artefact that you intend to produce?

A program that can transfer files from one computer across a LAN network to multiple computers

2.2 How is your artefact actionable (i.e., routes to exploitation in the technology domain)?

The artefact to this program is going to a real world application that could be distributed and used by others

Section 3: Evaluation

3.1 How are you going to evaluate your work?

I am going to be creating and developing the artefact while adhering to the DSDM development methodology. My evaluation will be on the effectiveness of DSDM as a development technique, my experiences with it, how it fits the scope of this project, ultimately how useful it was when developing the artefact and of course things that I would have done differently and how I would improve or what I would do again given a second chance

3.2 Why is this project honours worthy?

This project encapsulates core activities that I want to be pursuing in my computing career, it is a fully fledged development project for a real world company that requires this tool to work effectively. From a practical stand point I will have to develop a program that can be professional used by the client; it has to be functional (containing all the functionality of the clients previous program and requested improvements), stable and reliable as the tool is a time saver and generally a semi mission critical tool for the client. Technically this requires me to write my own network and transfer code to reliably transfer files with consistency and speed as this is going to be used in a professional and of course time sensitive

environment. I believe that a majority of the coding I do, if it works correctly, won't even be noticed. This is a real world development project, it specialises in networking and programming both areas I am very passionate about have been studying for years. It is going to be a stellar example of what I hope to achieve with my art and all the skills that I have learnt over my honours course and what direction I want to go with both my life and my career

3.3 How does this project relate to your degree title outcomes?

My project encapsulates everything that I wanted to achieve when I applied for to do a Bcs Computing degree. It will show off how much I have learnt about applications, usability and programming that I have learnt on this course on top of my previous experience. And the course has given me insight into many other areas that I had never considered before and are going to be exceedingly useful in my project and my future career such as client needs and program usability.

3.4 How does your project meet the BCS Undergraduate Project Requirements?

The project meets undergraduate requirements as it will require me to do background and contextual research into the problem of my project, abide by the system development stages of my chosen methodology and produce a program to good practices and standards

3.5 What are the risks in this project and how are you going to manage them?

One of the risk with the project is the fact that the program may not be entirely feature complete by the end of the project, one way I hope to mitigate this is by handing over all my research to the client after so that future upgrades and changes can be done one the basis of strong research

Section 4: References

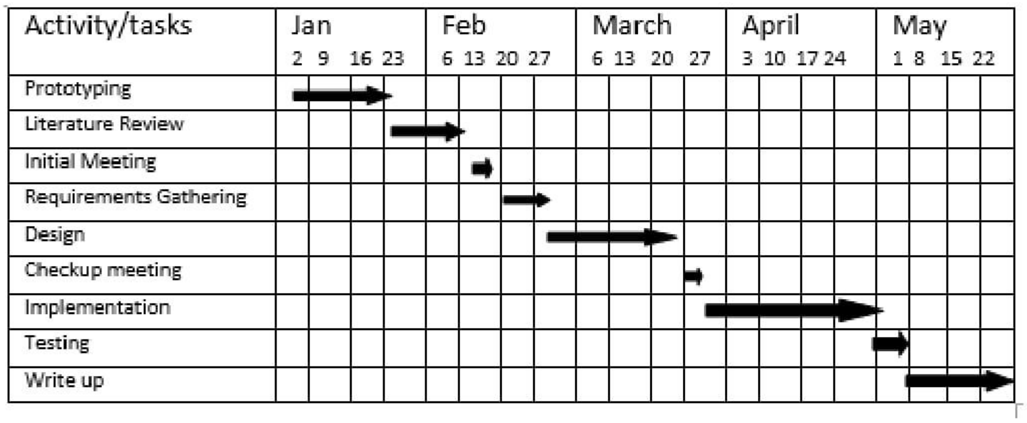
4.1 Please provide references if you have used any.

Section 5: Ethics (please delete as appropriate)

5.1 Have you submitted the ethics checklist to your supervisor? **Yes**

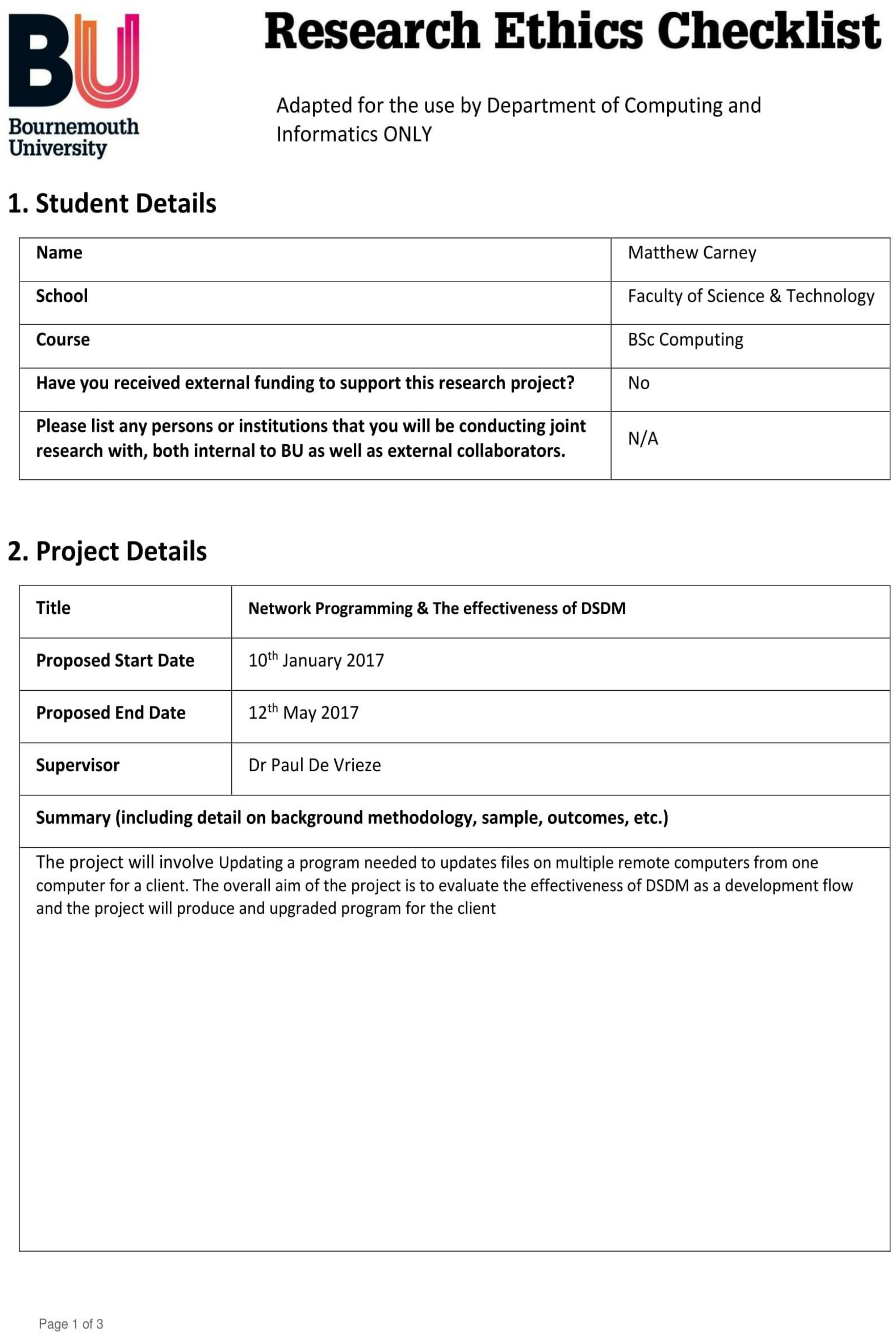
5.2 Has the checklist been approved by your supervisor? **Yes**

Section 6: Proposed Plan (please attach your <u>Gantt chart</u> below)



Activity/tasks	Jan				Feb				March				April				May			
	2	9	16	23	6	13	20	27	6	13	20	27	3	10	17	24	1	8	15	22
Prototyping	➡	➡	➡																	
Literature Review				➡	➡															
Initial Meeting						➡														
Requirements Gathering							➡													
Design								➡	➡	➡	➡									
Checkup meeting												➡								
Implementation												➡	➡	➡	➡	➡				
Testing																	➡			
Write up																		➡	➡	➡

APPENDIX C – ETHICS CHECKLIST



Bournemouth University

Research Ethics Checklist

Adapted for the use by Department of Computing and Informatics ONLY

1. Student Details

Name	Matthew Carney
School	Faculty of Science & Technology
Course	BSc Computing
Have you received external funding to support this research project?	No
Please list any persons or institutions that you will be conducting joint research with, both internal to BU as well as external collaborators.	N/A

2. Project Details

Title	**Network Programming & The effectiveness of DSDM**
Proposed Start Date	10th January 2017
Proposed End Date	12th May 2017
Supervisor	Dr Paul De Vrieze

Summary (including detail on background methodology, sample, outcomes, etc.)

The project will involve Updating a program needed to updates files on multiple remote computers from one computer for a client. The overall aim of the project is to evaluate the effectiveness of DSDM as a development flow and the project will produce and upgraded program for the client

Bournemouth University

Research Ethics Checklist

Adapted for the use by Department of Computing and Informatics ONLY

3. External Ethics Review (Answer "Yes" go to 4, "No" go to 5)

Does your research require external review through the NHS National Research Ethics Service (NRES) or through another external Ethics Committee?	No

4. External Ethics Review Continued

Answered "Yes" to question 3 will conclude the BU Ethics Review so you do not need to answer the following questions. Note you will need to obtain external ethical approval before commencing your research.

5. Research Literature (Answer "Yes" go to 6, "No" go to 7)

Is your research solely literature based?	No

6. Research Literature Continued (Either answer will conclude the review)

Will you have access to personal data that allows you to identify individuals OR access to confidential corporate or company data (that is not covered by confidentiality terms within an agreement or by a separate confidentiality agreement)?	Choose an item.
Describe how you will collect, manage and store the personal data (taking into consideration the Data Protection Act and the Data Protection Principles).	

Research Ethics Checklist

Adapted for the use by Department of Computing and Informatics ONLY

7. Human Participants Part 1 (Answer "Yes" go to 8, "No" go to 12)

Will your research project involve interaction with human participants as primary sources of data (e.g. interview, observation, original survey)?	Yes

8. Human Participants Part 2 (Answer **any** "Yes" go to 9)

Does your research specifically involve participants who are considered vulnerable (i.e. children, those with cognitive impairment, those in unequal relationships—such as your own students, prison inmates, etc.)?	No
Does the study involve participants age 16 or over who are unable to give informed consent (i.e. people with learning disabilities)? NOTE: All research that falls under the auspices of the Mental Capacity Act 2005 must be reviewed by NHS NRES.	No
Will the study require the co-operation of a gatekeeper for initial access to the groups or individuals to be recruited? (i.e. students at school, members of self-help group, residents of Nursing home?)	No
Will it be necessary for participants to take part in your study without their knowledge and consent at the time (i.e. covert observation of people in non-public places)?	No
Will the study involve discussion of sensitive topics (i.e. sexual activity, drug use, criminal activity)?	No

9. Human Participants Part 2 Continued

Describe how you will deal with the ethical issues with human participants?
I will be interacting and conducting my reseach into the clients using the techniques laid out by the DSDM Agile development flow

Research Ethics Checklist

Adapted for the use by Department of Computing and Informatics ONLY

10. Human Participants Part 3 (Answer **any** “Yes” go to 11, **all** “No” go to 12)

Could your research induce psychological stress or anxiety, cause harm or have negative consequences for the participant or researcher (beyond the risks encountered in normal life)?	No
Will your research involve prolonged or repetitive testing?	No
Will the research involve the collection of audio materials?	No
Will your research involve the collection of photographic or video materials?	No
Will financial or other inducements (other than reasonable expenses and compensation for time) be offered to participants?	No

11. Human Participants Part 3 Continued

Please explain below why your research project involves the above mentioned criteria (be sure to explain why the sensitive criterion is essential to your project’s success). Give a summary of the ethical issues and any action that will be taken to address these. Explain how you will obtain informed consent (and from whom) and how you will inform the participant(s) about the research project (i.e. participant information sheet). A sample consent form and participant information sheet can be found on the Research Ethics website.

Bournemouth University

Research Ethics Checklist

Adapted for the use by Department of Computing and Informatics ONLY

12. Final Review

Will you have access to personal data that allows you to identify individuals OR access to confidential corporate or company data (that is not covered by confidentiality terms within an agreement or by a separate confidentiality agreement)?	Yes
Will your research take place outside the UK (including any and all stages of research: collection, storage, analysis, etc.)?	No
Please use the below text box to highlight any other ethical concerns or risks that may arise during your research that have not been covered in this form.	

Review Completion Date: 11-May-2017 – Double click to change it!

The following section is to be filled by the supervisor only

Supervisor's Review:

Choose an item.

Please leave your comments:

Bournemouth University

Research Ethics Checklist

Adapted for the use by Department of Computing and Informatics ONLY

Help

Q3 External Ethics Review

If you choose "Yes", it will **conclude your BU Ethics Review**. Please ensure you obtain external ethical approval before commencing your research. Contact the Dorset Research Consortium with any questions regarding NRES application.

Q4 Research Literature

Please be careful when choosing "Yes" as this means your research will only use publicly available data (e.g., research paper, market reports) or permitted private/confidential data. This also means there will be no human participants involved in your project in any form (e.g., no client, no user testing, no interview, no focus group, no survey, no field study, no observation).

Q5 Research Literature Continued

Choosing either "Yes" or "No" will end this review BUT you will need to fill in the following field.

Q7 Human Participants Part 1

Note: please be careful when choosing "No" as this means there will be no human participants involved in your project in any form (e.g., no client, no user testing, no interview, no focus group, no survey, no field study, no observation).

Above minimal risk

The "Above minimal risk" will be automatically identified if you answered yes to one or more questions below. In that case, you **must be careful**. Consult your supervisor or project tutor if you need help. Alternatively, you may contact Sarah Bell, RKEO Research Governance Adviser at sarah.bell@bournemouth.ac.uk.

- Does your research specifically involve participants who are considered vulnerable?
- Does the study involve participants age 16 or over who are unable to give informed consent?
- Will the study require the co-operation of a gatekeeper?
- Will it be necessary for participants to take part in your study without their knowledge and consent?
- Will the study involve discussion of sensitive topics?
- Could your research induce psychological stress or anxiety, cause harm or negative consequences for the participant?
- Will your research involve prolonged or repetitive testing?
- Will the research involve the collection of audio materials?
- Will the research involve the collection of photographic or video materials?
- Will financial or other inducements be offered to participants?
- Will your research take place outside the UK?