

.NET Web Services Version Management System Initial Document

Dafydd Vaughan
327039
dafydd@dafyddvaughan.co.uk

October 20, 2006

Abstract

Keeping track of changes and progress of projects where there may be hundreds of people making changes at the same time is a very difficult task, but it is also an increasing problem for organisations. Further complications can occur when some users may be using different operating systems to others, making the sharing of files difficult. This project will look into using a new technology called Web Services to develop a solution to this problem. In this initial document I will cover the background to the problem and what technologies exist to solve the issue.

Contents

1	Introduction	4
1.1	Motivation	4
1.2	Aims of Project	4
2	Background	5
2.1	Version Control	5
2.1.1	What are Version Control Systems?	5
2.1.2	Existing Version Control Systems	5
2.1.3	How Do Version Control Systems Work?	7
2.2	Web Services	8
2.2.1	What are Web Services?	8
2.2.2	Other Distributed Technologies	9
2.2.3	Underlying Technologies	12
2.3	.NET Framework	16
3	Requirements Specification	18
3.1	Essential Features	18
3.2	Extra Features	18
3.3	Future Expansion / Upgrades	19
3.4	Notes	19
4	Planning	20
4.1	Milestones	20
4.2	Timeline	20
	Glossary	21
	References	23

Outline

This document is an introduction to a dissertation project that will aim to produce a working version management system using Web Services and the .NET framework. The document is split into 5 sections:

- **Section 1 - Introduction** looks at the motivation and aims behind this dissertation project
- **Section 2 - Background** looks deeper at the technical background of version control systems including CVS, SVN and SourceSafe, as well as an indepth look at distributed programming techniques such as COM/D-COM, CORBA, JAVA RMI, XML-RPC, Web Services and the new .NET Framework. This section examines the past and present technologies used to produce programmes that communicate with each other efficiently.
- **Section 3 and 4 - Requirements Specification and Planning** details the features that will be included in this dissertation project and the timeline that will be followed throughout the duration of the project.
- **Section 5 - Glossary** contains a list of the acronyms used throughout the document together with their meanings.

1 Introduction

1.1 Motivation

In many aspects of work, it is necessary to create and maintain documents collaboratively, whether it be a report, or a software project worked on by one person or an entire organisation. In all of these cases, it is important to successfully manage document changes. Keeping track of changes, versions and who is working on what is of key concern to many businesses.

This is where Version Control Systems come in. Version Control Systems (VCS) allow businesses, developers and individuals to keep track of changes to their files. Most VCSs will keep track of changes, and while some allowing them to be rolled back, some come with desktop applications whereas others use a website interface. Each VCS has a different set of features and does things differently. However, they all have one thing in common - they are platform specific. This means that they only work with certain set ups - i.e. specific operating systems. Some VCSs can also be difficult to set up.

A way of resolving these issues would be to use a new set of technologies called Web Services. The technologies used in Web Services make communication between programs on different platforms easy. By using open standards, Web Services can be implemented on any operating system, using any programming language.

1.2 Aims of Project

The aim of this project is to resolve the issues mentioned above and create a prototype tool that can be implemented on many different platforms. This project will make use of Web Services and the Microsoft .NET Framework to achieve this goal. The prototype developed will have an intuitive user interface as well as an easy to use API (Application Programming Interface) to enable other applications to be developed to use the service.

2 Background

2.1 Version Control

2.1.1 What are Version Control Systems?

Version Control Systems are applications that record changes made to a project, whether it be changes to one document, or hundreds[1]. Many version control systems allow users to see when changes were made to a document and what these changes were - this allows organisations to keep track of how the project is progressing. A version control system usually consists of a ‘server’ program that maintains the database of changes, and a ‘client’ program that allows the users access to the files and records. In many cases this client program is a website interface.

Although version control systems are generally associated with software programming, they can actually be used for any project that involves documents that are being changed regularly or being worked on by many people.

There are many different version control systems in use around the world. In the next section, we will look at some of these and examine their advantages and disadvantages.

2.1.2 Existing Version Control Systems

There are many different systems available to users wishing to implement version control for their projects. The majority of these systems are open source - which means they can be downloaded and used for free; however there are also a number of paid-for systems that can be used as well. In this section, we will look at several open source systems and the features they provide.

CVS (Concurrent Versions System)

CVS or Concurrent Versions System is a Linux based version control system and is probably the most well known package currently in use. The CVS development is based on an older system called RCS (Revision Control System) which was used to manage versioning of a single file. CVS uses and extends RCS to be able to manage an entire project[1]. This application, like most version control systems, uses a client-server architecture. A server program holds a “repository” that contains all past and present versions of each document and maintains the record of changes. A client program holds a “working copy” of the files which the user can work on. When he has finished the required changes, they are then merged with the repository held on the server.

CVS allows multiple users to work on a project at the same time, merging the

changes into one version in the repository. As all previous versions are kept, users can look back and see when and where changes were made. CVS also allows multiple users to work on the same file at the same time. This has the potential to cause problems, as one user could overwrite the other's changes - however CVS employs a technique whereby changes are merged together. How this system works will be covered in the next section (2.1.3).

The biggest problem with CVS is that it is mainly a Linux based package. Although there are a few versions available for other operating systems such as Microsoft's Windows, most of the development work takes place for the Linux distribution. This also means that the majority of support and tutorials available are Linux based and do not cover use of CVS on other operating systems. Other problems with CVS stem from its RCS base. Unlike other version control systems, CVS assigns revision numbers to individual files instead of the whole project, and it also does not handle folder changes easily.

Further information on CVS can be found at <http://www.cvshome.org>.

SVN (Subversion)

SVN is another open source version management system. SVN was developed as a replacement for CVS as some developers considered it difficult to use and lacked some required features. The SVN project aimed to replicate all of the features available to users of CVS, while making improvements and adding extra features that would be useful to developers[2]. Ironically, until SVN became capable of managing projects by itself, CVS was used to develop the software.

Like CVS, SVN uses a client-server model. The server holds a repository of files, including any associated metadata (extra information such as what changes have been made, who made them and when they were made etc.). A client program holds a working copy of the files that the user can edit and make changes to. Unlike CVS however, revision numbers (the number that tells the user what version the file is) are not associated with individual files, but with the project as a whole. For example, assume a project has 3 files - A, B and C - each are brand new and have the revision number 0. In CVS, when a user updates file A and merges it with the server, it increments the revision number of this file, but not the others. This means the files have the revision numbers 1, 0 and 0 respectively. In SVN however, when the user updates file A, it updates the revision numbers of the entire project. This means that all the files gain the revision number 1. This difference means that SVN can handle filename and folder changes much more easily than CVS.

Other advantages of SVN include versions of the package for many different operating systems, and "atomic commits". This means that should a failure occur during an update, the update will be cancelled, reducing the chances of corruption and crashes.

Further information on SVN can be found at <http://subversion.tigris.org>.

SourceSafe

Unlike the previous version control systems like CVS and SVN, SourceSafe is not open source software. SourceSafe is developed by Microsoft and included in their Visual Studio IDE (Integrated Development Environment). This makes it a very useful version control system for developers building applications using the Microsoft .NET Framework.

Unlike SVN and CVS however, SourceSafe does not allow more than one person to make changes to a file at the same time. When a file is “checked out” (downloaded to his/her working directory), it is locked so no other users can make any changes. Only when the user uploads the changed version is it unlocked and made available again. This has the potential to cause issues in the development process - such as users checking out files and then forgetting to unlock them. These issues will be covered in more detail in section 2.1.3.

More information on SourceSafe can be found at <http://msdn.microsoft.com/ssafe>.

2.1.3 How Do Version Control Systems Work?

One of the biggest problems faced by version control systems is how to handle multiple users wanting to manage the same file at the same time[3]. There are several methods of dealing with this problem, one is to avoid the issue and not let more than one user edit the file at any one time. This is called the ‘Lock-Modify-Unlock’ solution.

In the Lock-Modify-Unlock solution, user A “checks out” a file (downloads a copy to edit), locking the file contained on the server in the process. When user B or C attempts to access the file, they are told that the file is locked and are not allowed to make any changes. Once user A has finished making changes, he merges it with the server which unlocks the file and allows user B to access it. Although this solution is the simplest, it has the potential to cause other problems. For example, if user A locks a file and then forgets about it, user B is unable to access it until either user A remembers or an administrator releases the lock. This solution is used in Microsoft’s SourceSafe (Section 2.1.2).

The other method of handling this problem is the Copy-Modify-Merge solution that is employed in SVN and CVS. In this solution, both user A and user B can download editable copies of the file. When user A finishes editing the file, he uploads it to the server which updates the version held in the repository. When user B attempts to upload his changes, the server notices that the version it holds has been updated since user B downloaded a copy and attempts to merge

them. If it cannot do this because, for example, the changes overlap, the server then asks user B to resolve the conflict. Although this solution can lead to some conflicts, it is likely that users will be editing different parts of the file and as such, the efficiency this provides outweighs the time taken to resolve the conflicts.

Although CVS and SVN employ the Copy-Modify-Merge solution, they also offer the ability to occasionally use Lock-Modify-Unlock on a file for cases where it is essential.

2.2 Web Services

As computer systems evolve, it has become more and more necessary for computer applications to communicate with each other and share data. This could be something simple such as a web-based calendar, allowing users to access their diary from anywhere in the world, through businesses storing stock information in a central location, to a business-to-business system allowing partners to conduct transactions between their two different systems. It is possible that this 'shared' data could be held on the same machine - however, it is more likely that they will be on different computers - potentially on opposite sides of the globe. Web Services are a new combination of technologies that enable communication between these systems.

2.2.1 What are Web Services?

Web Services are web-based applications which use open standards to exchange data between clients[4]. Web Services make use of established Internet standards such as HTTP, TCP/IP and XML, as well as using some of its own standards such as SOAP, WSDL and UDDI. Each of these standards play a different part in enabling communication between two systems. For example, XML (Extensible Markup Language) is used as a common language between two computers that could be using a different dialect, and WSDL (Web Service Description Language) is a way of describing what a Web Service can do. Each of these standards and what they all do will be covered in section 2.2.3. Open standards have been used to ensure that Web Services can be implemented across as many different platforms as possible, but more importantly, it allows methods to be invoked across platforms - i.e. a program on a Microsoft Windows system invoking a method on a Linux system.

In essence, Web Services are comprised of well defined programming interfaces - allowing them to be integrated into many different programs[5]. These interfaces are outlined in the description of the Web Services. This means the programmer can easily call and invoke the required methods and functions when integrating the Web Service.

2.2.2 Other Distributed Technologies

Since its inception, the Internet has primarily been used for browsing HTML pages and sending emails[6]. However, more recently, it has become more important to share information between many different systems. Over the last few years, many new technologies have been developed - some more successful than others - to assist in this communication. Despite some of these technologies being more successful than others, they all suffer from the same downfall - cross platform integration. Take, for example, a computer running the operating system Microsoft Windows, and a computer running a form of Unix. These systems both speak different “languages” and handle things in different ways. As such, a service using Microsoft’s DCOM technology is unable to communicate with a service written in the Object Management Group’s CORBA technology. This makes it extremely difficult to communicate between many different systems. Most of the existing technologies currently available go some way to solving the communication barrier between systems, however, each have their own disadvantages which have lead to alternatives being created to overcome them.

In this section we will look at some of these rival technologies, including:

- COM/DCOM
- CORBA/IIOP
- Java RMI
- XML-RPC

COM/DCOM

In the early 1990’s, Microsoft developed the Component Object Model (COM) and integrated it into its Windows operating system[7]. COM allowed programs to communicate with other applications on the computer. This meant that application developers could create components for software that could be re-used and combined with other components to create their applications.

As networking systems became more popular, it became more important to share components over several computers. With this in mind, Microsoft extended this service and added DCOM (Distributed COM). DCOM allowed the developers to execute components on remote machines as if it was on the same machine.

The biggest problem with COM and DCOM was that it made use of ‘keep-alive’ messages to ensure connections between machines were kept active. If used on a bigger scale - such as the Internet - this would have a marked effect on the network because of the sheer amount of traffic generated. Another downside

of the technology was that it was integrated into Windows, which meant that it could only be used on machines running that operating system, therefore limiting its usefulness.

Although COM and DCOM are still included in its operating systems (including the new Windows Vista), Microsoft recommends that developers use the new .NET Framework instead. The .NET Framework will be covered in more detail in section 2.3.

Further information on COM/DCOM can be found at <http://www.microsoft.com/com/>.

CORBA/IIOP

The Common Object Request Broker Architecture (CORBA) is much like Microsoft's COM. It is a standard for creating Object Request Brokers (ORBs). Clients wishing to invoke methods on a remote machine notify the appropriate ORB, identifying which methods it wants to invoke. The ORB then handles the request for the client. Like COM, with the rise of networking it became necessary to expand the technology to enable communication over the Internet. IIOP (Internet Inter-Orb Protocol) was developed as the protocol for communicating requests to ORBs over networks.

Unlike COM which is a proprietary technology, CORBA is an open standard managed by the OMG (Object Management Group) - an organisation made up of over 500 member companies[8]. While this means it is widely supported, it also means there are many different implementations of CORBA - each slightly different from the others. This, together with the fact that it is not supported on the Microsoft platform, has meant that it has been unable to achieve widespread adoption.

Further information on CORBA can be found at <http://www.corba.org>.

Java RMI

Java Remote Method Invocation (Java RMI) is much like CORBA in that it uses ORBs to handle communication. An RMI application consists of a server program which creates remote objects and then waits for clients to invoke their methods, and a client program, which invokes the methods[9]. RMI is integrated into the Java language, which means it is easy for Java developers to implement. However, as it relies on features of Java, it is not possible for it to be implemented in other languages. As Java has been designed to work on multiple platforms from Windows to UNIX and from desktops to handheld devices such as mobile phones, RMI applications can be used on most systems - making it a very useful system and outweighing the disadvantage of a set language.

Further information on Java RMI can be found at

<http://java.sun.com/products/jdk/rmi/>

XML-RPC

XML-RPC is possibly one of the most widely used and well known technologies currently in use. XML-RPC uses XML messages sent over the Internet to the remote machine. Responses from the remote machine are also sent back in XML. Code listing 1 below shows an example of an XML request to the server, and code listing 2 shows an example of the server's response. XML-RPC is widely used across the Internet for various applications - the most notable of which is weblogs. Many weblogs send XML-RPC "pings" to trackers to notify them that the blog has been updated so they can update their listing.

Listing 1: XML-RPC Request

```
<?xml version="1.0"?>
<methodCall>
  <methodName>test.getIPAddress</methodName>
  <params>
    <param>
      <value><string>cs.swan.ac.uk</string></value>
    </param>
  </params>
</methodCall>
```

In this listing, the local machine sends a message using XML to the server asking it for the IP Address of the computer called cs.swan.ac.uk.

Listing 2: XML-RPC Response

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>137.44.2.59</string></value>
  </param>
</params>
</methodResponse>
```

The server then replies with an XML message telling the client that the IP address is 137.44.2.59.

The XML-RPC specification is very small, which makes it an easy option to implement. As more features have been added to the specification, it has evolved into a new standard called SOAP (Simple Object Access Protocol) which is an integral part of Web Services. In fact, XML-RPC is considered to be a very simple Web Service. SOAP will be looked at in more detail in section 2.2.3. The biggest problem with XML-RPC, however, is its inability to handle the multiple data types required by more complicated systems - a problem resolved in SOAP.

More information on XML-RPC can be found at <http://www.xmlrpc.com>.

Web Services

Web Services have been developed by the W3C (World Wide Web Consortium) and supported by Microsoft, Sun and IBM - three of the biggest players in Internet technologies. This means that Web Services are “Vendor Neutral” and can be implemented by any platform. Web Services also differ from previous technologies in that they use multiple open standards to ensure it is completely platform independent - i.e. they can be implemented on any computer using any operating system. The biggest advantage of Web Services however is that they are easy to implement. Some of the previous technologies covered in this section are in decline because they are difficult to implement correctly.

Microsoft has also included Web Services within its new .NET Framework, enabling .NET developers to easily create Web Services for their applications. The biggest advantage of Web Services are that services developed in the .NET Framework can be used by non-.NET applications, meaning that they can be used by any application.

The underlying technologies used by Web Services will be covered in more detail in the next section (2.2.3).

2.2.3 Underlying Technologies

Unlike the previous technologies mentioned above, Web Services are a collection of different technologies that work together to make communication easy. The following section will outline some of the technologies used in Web Services, including:

- XML (Extensible Markup Language)
- HTTP (Hyper Text Transfer Protocol)
- SOAP (Simple Object Access Protocol)
- WSDL (Web Service Description Language)

- UDDI (Universal Discovery, Description, and Integration)
- TCP/IP (Transmission Control Protocol / Internet Protocol)

XML (Extensible Markup Language)

XML or Extensible Markup Language looks a lot like HTML (the language used for creating web pages). Like HTML, XML contains tags, attributes and values, but unlike HTML, it is not used for creating web pages. Instead, XML is a language for creating other languages [10]. The tags used in XML describe the data they contain. For example the code listing 1 in section 2.2.2 above, contains the tag:

```
<methodName>test.getIPAddress</methodName>
```

From this, you can see that the tag “methodName” contains the method/procedure to be used by the receiving server.

The advantage of XML over creating a completely new language is that all languages using XML as a base look similar to each other, so they are easier to learn. Also, XML is designed to be extremely flexible, making it simple to create a new language for a specific purpose.

XML is used as a base for many of the other standards used in Web Services, including WSDL (Web Service Description Language) and SOAP (Simple Object Access Protocol). These languages will be covered later in this section.

Further information on XML can be found at <http://www.w3.org/XML/>.

HTTP (Hypertext Transfer Protocol)

HTTP or Hypertext Transfer Protocol is the protocol used for communicating messages from a client to a server and vice versa over the Internet. Each communication consists of a request and a response [11]. For example, when you enter a URL (website address) into a web browser, it sends a HTTP request to the appropriate server and the server responds with the page you requested.

HTTP is the transport method of choice for Web Services. SOAP messages are sent over HTTP to the client and server. However HTTP is not particularly efficient, and as such, work is underway to enable Web Service messages to be sent over other protocols such as SMTP (Simple Mail Transfer Protocol).

Further information on HTTP can be found at <http://www.w3.org/Protocols/>.

SOAP (Simple Object Access Protocol)

SOAP or Simple Object Access Protocol is an XML based language that is used by Web Services to send messages to the server and back. Requests to the server are written in SOAP and sent over HTTP. As SOAP is based on XML, the messages consist of plain text, which means they can be used to send complex objects as well as standard strings (text) and integers (numbers) [12]. Another advantage of SOAP is that it can be passed through network firewalls easily because it uses the same port to communicate as web pages. This is in contrast to previous technologies such as IIOP and DCOM which could be caught and blocked by firewalls¹.

SOAP has evolved from the XML-style messages that became successful in XML-RPC.

Listing 3: SOAP Message

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/
  soap/envelope/">
  <soap:Body>
    <getProfile xmlns="http://studentrecords.
      college.ac.uk/ws">
      <studentID>327039</studentID>
    </getProfile>
  </soap:Body>
</soap:Envelope>
```

The code above (listing 3) is an example of a SOAP message sent to a college's student records database requesting details of the student with the student ID 327039. The code below (listing 4) is an example of the server's response.

Listing 4: SOAP Message

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org
  /soap/envelope/">
  <soap:Body>
    <getProfileResponse xmlns="http://studentrecords.
      college.ac.uk/ws">
```

¹It is possible however for some technologies, such as Java RMI to be tunneled to avoid firewall restrictions

```
<getProfileResult>
  <studentName>Dafydd Vaughan</studentName>
  <studentID>327039</studentID>
  <studentDept>Comp Sci</studentDept>
  <studentCourse>MEng Computing</studentCourse>
  <studentYear>3</studentYear>
</getProfileResult>
</getProfileResponse>
</soap:Body>
</soap:Envelope>
```

As SOAP messages are independent from the programming languages that the server and client applications are developed with, messages are completely platform independent. For example, a server written in C# on the Microsoft .NET platform can understand messages received from a client written in Java on a UNIX platform. This independence is what sets Web Services apart from other previous technologies.

Further information on SOAP is available at
<http://www.w3.org/TR/soap/>.

WSDL (Web Service Description Language)

WSDL or Web Service Description Language is a language for describing to clients what methods and functions are available for a particular Web Service[13]. The documents produced using WSDL also show what outputs can be expected when a method is activated, what values are needed to invoke the method and what data types are returned by the service.

Previous technologies such as COM/DCOM and CORBA/IIOP used a similar document to outline their available methods, however, WSDL documents are based on the XML language - like SOAP above. This means the documents are easy to understand. Also, like SOAP, the documents are independent from the language used to develop the client and server applications, enabling it to work on multiple platforms.

Further information on WSDL is available at
<http://www.w3.org/TR/wsdl/>.

UDDI (Universal Discovery, Description, and Integration)

UDDI or Universal Description, Discovery and Integration is a directory that lists available Web Services that can be used by developers. The UDDI directories enable clients to discover the Web Services they need for their applications easily, and like WSDL and SOAP, use an XML base to ensure they can be im-

plemented and understood on multiple platforms. UDDI directories can either contain services from multiple sources or just from one organisation. In addition, it may only be available from inside an organisation's own network. The aim of UDDI is to hold all available services in one location, to make it easier to reuse them.

Microsoft have also incorporated a UDDI directory system into it's new version of Windows Server 2003 to encourage businesses to develop their own Web Services.

Further information on UDDI is available at <http://www.uddi.org>.

TCP/IP (Transmission Communication Protocol / Internet Protocol)

TCP/IP or Transmission Control Protocol / Internet Protocol are the two protocols upon which the Internet and most networks operate/citebibtcpipdesc. These two protocols define how communications are transported over the network. Unlike HTTP, which defines how the messages are formed, TCP/IP defines how the messages are actually sent. All the technologies that form Web Services sit on top of the TCP/IP protocols and make use of its operations.

Summary

Web Services are formed of a large number of technologies that work together to make them compatible with as many platforms as possible. This is in contrast with previous technologies which have been platform specific. All the Web Services technologies above (SOAP, WSDL, UDDI) are independent from whichever language or platform is actually used to develop the server/client applications which ensures services written for one platform can still be used by others.

Below is a summary of all of these technologies and how they fit together.

TCP/IP	Handles how messages are sent over the network
HTTP	Handles how the messages are communicated (i.e. their format)
XML	The base language upon which the other technologies are based
SOAP	The language in which requests/responses are sent
WSDL	The language from which descriptions of Web Services are made
UDDI	A directory containing lists of available web services

2.3 .NET Framework

Microsoft's .NET Framework is a new software development platform for their Windows operating system. The .NET Framework consists of a library of code 'classes' and a 'Common Language Runtime' (CLR), much like the Java platform. The code library contains a wide range of methods, functions and data

types that are used most regularly by software developers, enabling them to easily reuse code. The CLR acts much like the Java Virtual Machine in that it creates a protected area from which the application is executed. It is a common misconception that the .NET Framework is a language - it is not. Applications that make use of the .NET Framework can be written in many different languages including C#, VB.NET and J#.

The .NET Framework has been developed as a replacement for Microsoft's older COM/DCOM, which were very difficult to use. The biggest advantage of the .NET Framework is that it is interoperable - which means it makes use of both new code libraries and older COM code libraries. Since its first release in 2001, the .NET Framework has gone through many updates, including improving security and adding new libraries. The latest stable version (2.0) has introduced Web Services, making it even easier for programmers to develop distributed software.

More information on the .NET Framework is available at <http://www.microsoft.com/net>.

3 Requirements Specification

This section outlines the features I will be aiming to implement into the .NET Web Services Version Control System. The features list is split into 3 categories:

- **Essential Features** - functionality that must be included for the project to be considered a success
- **Extra Features** - functionality I would like to have completed but are not essential
- **Future Expansion / Updates** - functionality that could be added in the future

3.1 Essential Features

The features included within this section are categorised “essential” and need to be completed for the project to be considered a success.

1. A Web Service “server” that:
 - allows a client to see files contained within the project
 - allows a client to “check out” (download) files to the local machine
 - updates revision number on commit (upload)
 - checks version number of clients copy against server copy on commit
 - merges changes when revision numbers do not match
2. A Web Service “client” that:
 - downloads a copy of a file to the local machine
 - allows the user to “commit” their copy to the server

3.2 Extra Features

The features included within this section are features that I would like to be completed by the end of the project, however they are not considered essential.

1. A server that maintains previous versions of a project and enables the project to be rolled back
2. A server that maintains a log of all changes for auditing purposes

3. A server that allows files to be “locked” from edit when necessary (following the Lock-Modify-Unlock solution to multiple user edits)
4. A server with a web interface for easy management
5. A server and client with in built authentication
6. A client that allows files to be downloaded if the server copy has been updated since download
7. A Java based client for communicating with the server to test out the new version of the Java platform

3.3 Future Expansion / Upgrades

The features included within this section are features that could be added to the application in the future to enable greater functionality for users. These future features could include:

- the ability to branch the project to enable 2 parallel projects
- a server application written in the new Java platform
- a bug tracking system built into the server and client

3.4 Notes

When starting the project, I decided to develop the version control system using Web Services on the .NET Framework as this was the most straight forward way of building them. However, since then, Sun have released a new beta version of Java which contains support for Web Services. As a result, I have added an additional feature into section 3.2 and aim to develop a simple client using this new version of Java.

For the essential functionality I will be using the C# language.

4 Planning

4.1 Milestones

There are several important milestones that mark important progress points in this project. These milestones include:

- **Initial Document Completion (20th October 2006)** - at this point a full feature list will have been created and the designing of the final software will be able to commence
- **Web Service Experiment Completion (15th December 2006)** - at this point, I will have achieved a greater understanding of Web Services and how they will be used to create the final client and server applications
- **Prototype Build Completion (15th January 2007)** - at this point, a prototype of the server application will have been created
- **Essential Functionality Completion (19th March 2007)** - at this point, a beta version of the server and client will have been created, enabling testing to commence and the development of additional features
- **Release Candidate 1 (15th Apr 2006)** - at this point, the development and testing will have been completed
- **Project Completion (4th May 2006)** - at this point, the project will be complete

4.2 Timeline

Date	Details
29th August 2006	Background research of Web Services, version control systems and the .NET Framework begins
20th October 2006	Initial document complete
23rd October 2006	Web Service experiments begin
20th November 2006	Project presentation at the Undergraduate Computer Science Colloquium at Gregynog
15th December 2006	Web Service experiments complete
18th December 2006	Timeline review and development of key features begins
15th January 2007	Prototype of server application complete
8th February 2007	Interim document complete
19th March 2007	Essential features completed
19th March 2007	Testing and extra feature development begins
15th April 2007	Feature complete release candidate available
4th May 2007	Project completion

Glossary

.NET	Microsoft .NET Framework - Microsoft's new technology to enable greater communication between programs., 7, 10, 12
API	Application Programming Interface - an interface that allows developers to create compatible applications., 4
CLR	Common Language Runtime - the virtual machine used by the Microsoft .NET Framework., 16
COM	Component Object Model - an old Microsoft technology used to enable communication between programs., 9, 15
CORBA	Common Object Request Broker Architecture - a technology for enabling communication between programs., 8-10, 15
CVS	Concurrent Versioning System - an open source version management system developed for the Linux platform based on RCS., 5-7
DCOM	Distributed COM - an extension of COM that allows components to communicate with each other over a network., 8, 9, 13, 15
HTML	Hypertext Markup Language - a language for creating web pages., 8, 13
HTTP	Hypertext Transfer Protocol - a protocol used for managing communication of messages across the Internet., 8, 12, 13, 16
IDE	Integrated Development Environment - an application that allows developers to build software easily., 7
IIOP	Internet Inter-ORB Protocol - a protocol for enabling communication between programs over the Internet., 9, 10, 13, 15
IP Address	Internet Protocol Address - a unique address for a specific computer on a network., 11
Java RMI	Java Remote Method Invocation - a Java implementation of CORBA/IIOP. Integrated into the Java programming language., 9, 10

RCS	Revision Control System - an old open source version management system that manages versioning for single files., 5
SMTP	Simple Mail Transfer Protocol - the protocol that is used to handle the sending of emails., 13
SOAP	Simple Object Access Protocol - a language used to send Web Service messages., 8, 12, 13, 15, 16
SourceSafe	Source Safe - a paid-for version management system developed by Microsoft and integrated into their IDE., 7
SVN	Subversion - a new open source version management system built to replace CVS., 6, 7
TCP/IP	Transport Communication Protocol / Internet Protocol - a set of protocols used for managing the transport of messages over a network., 8, 12, 16
UDDI	Universal Discovery, Description and Integration - a directory used to store details of available Web Services., 8, 12, 15, 16
URL	Uniform Resource Locator - a name that points to the location of a web page., 13
WSDL	Web Service Description Language - a language used to describe a Web Service and its features., 8, 12, 15, 16
XML	Extensible Markup Language - a flexible HTML like language that is used for creating other languages., 8, 11–13, 15
XML-RPC	XML Report Procedure Call - A technology used to send primitive messages from one service to another across the Internet., 9, 11, 14

References

- [1] J. Vesperman, *Essential CVS*, O'Reilly, 2003 pp 3–5.
- [2] B. Collins-Sussman, B. Fitzpatrick, C. Pilato, *Version Control with Subversion*, O'Reilly, 2004 pp 1–2.
- [3] B. Collins-Sussman, B. Fitzpatrick, C. Pilato, *Version Control with Subversion*, O'Reilly, 2004 pp 9–13.
- [4] A. Freeman & A. Jones, *Microsoft .NET XML Web Services Step By Step*, Microsoft Press, 2003 pp 3.
- [5] T. Shelford & G. Remillard, *Real Web Project Management*, Addison-Wesley, 2003 pp 271–272.
- [6] A. Ferrara & M. MacDonald, *Programming .NET Web Services*, O'Reilly, 2002 pp 4–9.
- [7] A. Ferrara & M. MacDonald, *Programming .NET Web Services*, O'Reilly, 2002 pp 5–6.
- [8] A. Ferrara & M. MacDonald, *Programming .NET Web Services*, O'Reilly, 2002 pp 7–8.
- [9] Java Remote Method Invocation
<http://java.sun.com/products/jdk/rmi/>
Accessed 18th October 2006.
- [10] E. Castro, *XML for the World Wide Web*, Peachpit Press, 2001 pp 11–13.
- [11] A. Tanenbaum, *Computer Networks (Fourth Edition)*, Pearson Education, 2003 pp 651–652.
- [12] C. Payne, *Teach Yourself ASP.NET*, Sams Publishing, 2003 pp 576.
- [13] A. Ferrara & M. MacDonald, *Programming .NET Web Services*, O'Reilly, 2002 pp 12.
- [14] A. Tanenbaum, *Computer Networks (Fourth Edition)*, Pearson Education, 2003 pp 532–556.