Final-Project Report
Calendar Tool
Shengqun Sun
Computer Science
2002/2003

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student) _____

Summary

Before 1980s, the personal computers are not usually used by people. People have to record information about events on paper or remember them in mind, and if they want to schedule a meeting, they have to compare the timetables of attendees or even have to make a phone call to all attendees to check suitable meeting time. So that wastes not only labor power and also material resources. The efficiency of working was decreasing a lot.

Advancement in computer engineering and information technology has reached a state in the past decade that leads to the widespread ownership of personal computers worldwide, with an ever-increasing population of internet users, carrying out numerous types of transactions online. Thus some calendar tools produced are used to help people ensure that information can be quickly and reliably stored and organized. What's more, people can exchange calendaring and scheduling information in an easy and automated manner.

It is the objective of this project to developing an all-in-one software that is able to store information about events and handle information in many data formats. Program can exchange important data about events so that users can schedule meetings with anyone else who has a same date format aware program.

Acknowledgements

I would like to offer my sincere thanks to my supervisor Dr. Nick Efford, for his guidance throughout the planning and execution stages of this project, whose constant advice and helpfulness are invaluable to my project, and also thank my assessor Natasha Shakhlevich, for her advice on writing up the report. The successful implementation of Calendar Tool program would not have been possible if without their patience and continual support.

I also would like to thank many friends who have annoyed greatly with testing of my system.

Finally, I would like to thank my family for their continuous support, love and enthusiasm, which has always accompanied me particularly in the difficult times.

Contents

Chapter 1

Introduction

1.1 Purpose

This project is concerned with building and programming a Calendar Tool so that users can view and manage their appointments quickly, reliably and electronically. This program can exchange important data about events so that people can schedule meetings with anyone who is using a similar kind of program.

1.2 Similar Software Product

Here I introduce two similar software products so that the reader can further understand the problem of this project.

1. Ximian Evolution Calendar in Linux Environment.
2. Outlook Calendar in Window Environment.

1.2.1 Ximian Evolution Calendar in Linux System

Ximian Evolution Calendar is a calendar and scheduling application used in Linux Environment, all calendar information is stored in a “calendar.ics” with iCalendar format. Ximian Evolution Calendar offers user four different views of calendar, which are day, working week, week and month respectively. Moreover, User can select an arbitrary range of days that user wishes to view in the calendar. The range of days can exceed a month.

Ximian Evolution Calendar also can add new appointments that must have a starting and ending date, but user can choose whether to give it starting time and ending time or to mark it as an all day event. An all day event will be displayed in grey area on the top before time 00:00 and under the current date rather than inside it, so appointments can overlap and fit inside each other. For example, a conference might be an all day event including some timed meeting, and the meetings would be timed appointments. Of course, appointments with specific starting time and ending time can also overlap. They will be displayed as multiple columns in the day view of the calendar. What's more, Ximian Evolution Calendar supports the use of time zones. If user wants to schedule a meeting, it is necessary to set time zone. User can also configure time zone information specific to the Start and End time in each appointment. For example, I live in UK but want to have a telephone meeting with my parents in China, the time difference is 7 hours, In order to avoid the potential confusion the time zone must be set for this appointment. In Addition user can use the Free/Busy view to check whether people are available in advance. The organizer can arrange a meeting by finding a common free time slot for all attendees.

The main window of Ximian Evolution Calendar is shown in figure 1.2.1 below to give reader some ideas what the user interfaces look like.

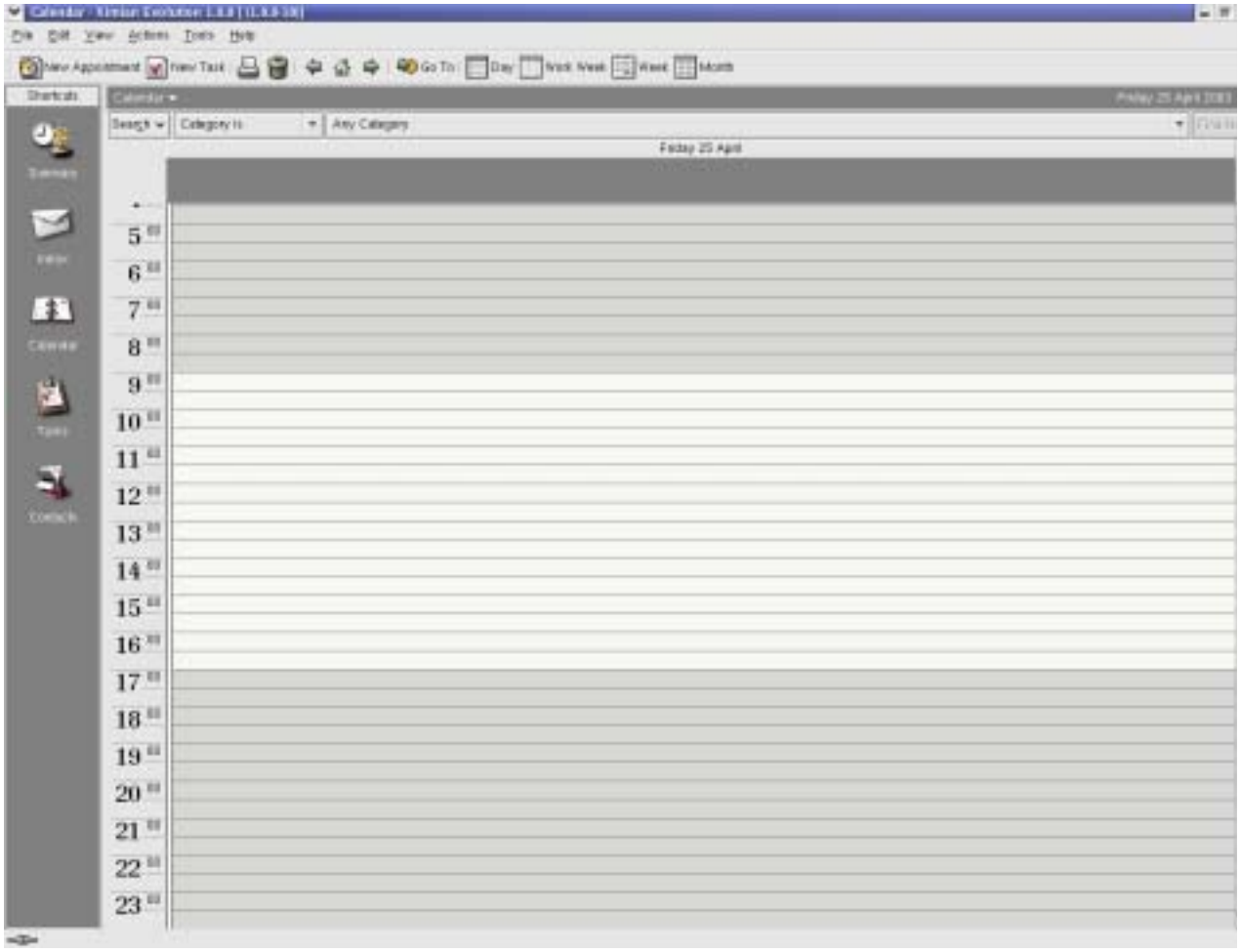

Figure 1.2.1 Main Window of Ximian Evolution Calendar

1.2.2 MSN Calendar in Window System

MSN Calendar can be found when MSN Explorer is started, which is used in the Windows environment. Functionalities of MSN Calendar are similar with Ximian Evolution Calendar. In addition, the MSN calendar can cooperate with MicroSoft outlook. For example, the MSN calendar can import appointments, tasks and notes up to 2000 entries from Microsoft outlook Calendar. The interface of MSN calendar is also different with Ximian Evolution Calendar and the main window of MSN Calendar is showed in figure 1.2.2 below:

Figure 1.2.2 Main Window of MSN Calendar

1.3 Minimum Requirement

1. Some fundamental functions must be included in the Calendar tool, such as adding a new event, deleting an event, viewing event(s), scheduling a meeting…

2. The Calendar can import files with one or more different standard calendar data formats, such as iCalendar, XML.

3. Understand how the Calendar tool could schedule a group of files and try to build it into Calendar.

Possible enhancement

Understanding the limitations of the Calendar Tool and how they can be improved.

1.4 Deliverables

1. Mid-Project Report

2. Final Calendar Tool Product

3. Final Project Report

1.5 Objectives

1. Fundamental functions of Calendar Tool should be sufficient for the tool to be used.

2. The calendar tool should also have ability to schedule a meeting with another who is using the similar kind of calendar tools.

3. The Calendar Tool should work efficiently and reliably, and be easy to use.

4. Finally, the calendar tool must be evaluated to understand its existing limitations.

1.6 Project Schedule

The table below briefly shows how the time was spent

Time	Month	Dec 2002				Jan 2003				Feb 2003				Mar 2003				Apr 2003			
	Week	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Background Reading																					
Design Calendar Tool																					
Programming Calendar to do the required task(s)																					
Testing as the Calendar is programmed																					
Modify the finished program and test again																					
Final write up																					

Chapter 2

Background Research

2.1 The Standard iCalendar File Format

Most calendaring and scheduling applications, such as Personal Information Manager or Group Scheduling Product, store their information in iCalendar format, because iCalendar format is more common and popular, and is also suitable as an interchange data stream format between applications or systems. Currently there is no particular method, which must be used to transport this kind of data format; however the file system may be used for exchange purpose. More important is that iCalendar is a transport and platform-independent format for exchanging calendaring and scheduling personal data.

2.1.1 iCalendar Object and vEvent Entity

An iCalendar is a data stream consisting of one or more iCalendar objects. Each individual iCalendar object can be identified and well formatted within the data stream. The iCalendar object seems like a container used to store calendaring and scheduling entities. An event is a calendaring and scheduling entity that represents an amount of specific time on a calendar, it may be an activity; such as attend an interview from 3 PM to 4 PM at University of Oxford, next Monday.

In an iCalendar format file, the string “BEGIN:VCALENDAR” must appear as the first line at

the start of data stream indicating that an iCalendar object is starting. When the delimiter string “END:VCALENDAR” appears, the iCalendar object terminates. All data between “BEGIN:VCALENDAR” and “END:ICALENDAR” belong to entities of iCalendar.

vEvent Entity is identified within an iCalendar object by the delimiter string “BEGIN:VEVENT”. A vEvent entity is terminated by the appearance of the End vEvent Delimiter string “END:VEVENT”. **[9]**

2.1.2 Syntax For Encoding the iCalendar

1. Date and Time

The values of date and time for iCalendar are represented as a string in format of <year><month><day>T<hour><minute><second>. For example, 3pm on March 15, 2003 would be written as 20030315T150000. We use Universal Coordinated Time (UTC), rather than local time that aims to avoid time zone ambiguities, so the above example in UTC based time would be written as 20030315T150000Z.

Created Date/Time is identified in the iCalendar file by name “DTSTAMP”

Start Date/Time is identified in the iCalendar file by name “DTSART”.

End Date/Time is identified in the iCalendar file by name “DTEND”.

2. Time Duration

Duration of time given by the user is represented as a string starting with the character “P”. There are some examples below to make this clear. “P3W” means a period of three weeks, “PT45M” means a period of 45 minutes, and “P1Y9M15DT10H30M10S” is a general case, which means a period of 1 year, 9 months, 15 days, 10 hours, 30 minutes, and 10 seconds.

3. Classification

‘Classification’ is an iCalendar property defining the access classification for the iCalendar entity, which is identified in the iCalendar file by the property name CLASS. The type of values of classification for iCalendar is strings. There are three property values for classification, which are public, private, and confidential. The default value for classification is public, the values are written in an iCalendar file as CLASS:PUBLIC.

4. Categories

‘Categories’ is a property defining the categories for the iCalendar entity, which is identified in the iCalendar file by the property name CATEGORIES. The type of values of categories for iCalendar is ‘string’. More than one category may be allowed for a vEvent entity separated by

the Semi-Colon character. For example, CATEGORIES:BUSINESS;EDUCATION.

5. Sequence

Sequence is a property defining the number of revisions that had been made to an iCalendar entity, which will be identified by the name SEQUENCE. When an iCalendar entity is created and its sequence is set to zero, this sequence will be incremented each time if the iCalendar entity is modified.

6. Summary

Summary is a property defining a short summary or subject of the iCalendar entity, which is identified by the name SUMMARY in the iCalendar file.
For example, SUMMARY: Shengqun Sun's report.

7.Time Transparency

Time Transparency is a property defining whether the event is transparent to free/busy time searches, which is identified by name TRANSP in the iCalendar file. If the value of this property is "TRANSPARENT", then specified time of this event is treated as free, otherwise busy. For example, TRANSP: TRANSPARENT

8. Unique Identifier

Unique identifier is identified by name UID, which defines a globally unique identifier associated with the iCalendar entity. Unique Identifier is an important property for group calendaring and scheduling applications to match calendar entities with later modification or deletion requests. It is optional to include this property to the implementation of iCalendar, but if a Calendaring application supports this property in iCalendar entities, it may be improving their interoperability with other group calendaring and scheduling applications. **[10]**

2.2 Differences between iCalendar and vCalendar

Both iCalendar and vCalendar are standard formats for storing calendar and todo data and under the control of the Internet Mail Consortium and specified in RFC 2445. In addition, iCalendar also is the base of the group scheduling standard iTIP in RFC 2446 and its implementation iMIP in RFC 2447.

vCalendar is the older version of the format and is currently supported by many products, especially use by some mobile phones and other mobile devices. iCalendar is a newer and more robust version of vCalendar that is capable of transferring more information. There are also numerous other applications, which can handle the iCalendar format, e.g. Evolution or Outlook. iCalendar files typically contain a line "VERSION:2.0" to distinguish them from the vCalendar format, which uses "VERSION:1.0".

In addition, there is a more obvious instance that shows the difference between iCalendar and vCalendar. The time transparency property is identified by name TRANSP in the iCalendar file with type of string. But the time transparency property is identified by name TRANSP in the iCalendar file with type of integer. The values of transparency property are specified in table below for both iCalendar and vCalendar.

Transparency property	Value in iCalendar	Value in vCalendar
Transparent	`TRANSP:TRANSPARENT`	`TRANSP:0`
Non-tansparent	`TRANSP:OPAQUE`	`TRANSP:1`

2.3 The XML File Format

2.3.1 XML Definition

Extensible Markup Language (XML) is a simple, very flexible text format. It was designed to meet the challenges of large-scale electronic publishing originally; XML is also playing an increasingly important role in the exchange of a wide variety of data on the web and between the different systems, and becoming more and more popular now.

XML was designed to describe data, and to focus on what data is. Tags are not predefined in XML, so we must define our own tags and our own document structure. **[1]** What's more, XML uses either a Document Type Definition (DTD) or an XML Schema to describe the data, and XML with a DTD or XML Schema is designed to be self-descriptive. **[2]**

2.3.2 Advantages of XML Format

In the real world, many computer systems and databases contain data in incompatible formats. Fortunately data can be exchanged between incompatible systems through using XML. Converting the data to XML can reduce the complexity of problem, and the data in XML also can be read by many different types of applications. So my Calendar Tool should have the ability to output personal information to a file with the standard XML, and also can convert the XML

format to a standard iCalendar format. **[2]**

2.3.3 Syntax in XML

1. Declarations

The first line in any XML file is XML declaration, which defines the XML version and the character encoding method currently used in an XML document. For example the line `<?xml version="1.0" encoding="iso-8859-1" >` means the document conforms to the 1.0 specification of XML and uses the ISO-8859-1 (Latin-1/West European) character set.

2. Elements

All XML elements must have a closing tag; it is illegal to omit the closing tag. What's more, XML tags are case sensitive; the tag <iCalendar> is different from the tag <icalendar>. In addition, XML elements have relationships. All elements are related as parents and children; **[3][4]**

3. Attributes

Attributes in XML provide additional information about the element, which is not a part of the data. What's more, all attribute values should always be enclosed in either single or double quotes. Double quotes are the most common, but sometimes single quotes must be used if the attribute value itself contains quotes such as `<game name ='arm core "another age"'>.`

2.3.4 Standard XML Documentation for Representing iCalendar (xCal)

1. Mapping calendar properties to XML

The standard features of the XML syntax can be applied to represent the component iCalendar semantics. For example, iCalendar data type can be specified using the XML notation declaration. The element type attributes in an iCalendar XML are used to represent many of the calendar properties that provide a "global attribute" capability in an iCalendar object. In the example below, it is easy to see that the method, version and prodid in XML representation become attributes of the "iCalendar" or the "vcalendar" elements.

2. Mapping component properties to XML

Component properties defined in the standard iCalendar format are represented in the XML

representation as element types. The table 2.3.4 below specifies the element types in XML corresponding to each component properties in iCalendar.

Component Property Name	Element Name
CATEGORIES	Categories
CLASS	Class
DESCRIPTION	Description
LOCATION	Location
SUMMARY	Summary
SEQUENCE	Sequence
DTSTAMP	Dtstamp
DTSTART	Dtstart
DTEND	Dtend
TRANSP	Transp
UID	Uid
VALARM	Valarm
TRIGGER	Trigger
VEVENT	Vevent

Table 2.3.4 Mapping iCalendar Component Properties to XML

3. The Standard iCalendar XML Documentation

A well-formed and standard iCalendar XML document is shown below to help people understand iCalendar XML quickly. The first line is XML declaration; the next line describes the root element of the document, which is telling us this XML document is a calendar file. The following lines describe some child elements of the root “calendar”, and finally the last line defines the end of the root element.

```
<?xml version=“1.0” encoding=“UTF-8” >
<iCalendar>
<vcalendar method=“PUBLISH” version=“2.0”prodid=“//Shengqun//SUN
Calendar//”>
<vevent>
      <uid>csxss003-425-811</uid>
      <dtstamp>20031009T233000Z</dtstamp>
      <dtstart>20031120T133000Z</dtstart>
      <dtend>20031122T183000Z</dtend>
      <summary>Calendar Tool</summary>
      <location>Level Ten Lab</location>
      <categories>
            <item>Education</item>
      </categories>
</vevent>
</vcalendar>
</iCalendar>
```

4. Multiple iCalendar Objects

The iCalendar format has the capability for including multiple individual iCalendar objects in a single data stream. The XML representation also supports this. Each Individual iCalendar object is identified by "vcalendar" element type, so more than one "vcalendar" element types are allowed within the single iCalendar element type. **[18]** For example:

```
<iCalendar>
       <vcalendar version="1.0">
              Here is the first iCalendar object
       </vcalendar>
       <vcalendar version="1.0">
              Here is the the second iCalendar object
       </vcalendar>
</iCalendar>
```

Altogether, XML is more flexible and widely used. Using the XML data format will increase the compatibility of my Calendar Tool with other similar kinds of applications.

2.4 Scheduling Problem

In order to schedule an appointment for a group of people, organisers need to know the free/busy information of each person. If the calendar system cannot connect to an exchange server at that moment, organisers will have to ask for schedule files from each attendee, then the scheduling application reads the information of all the received scheduling files, and starts scheduling; a common free time slot for all attendees will be found after checking, the busy time of each attendee will be filtered out during the checking. So it is sure that the determined time is suitable to everybody. In order to let everybody know the time of a meeting, the organiser is required to broadcast e-mail with an iCalendar or Xml attachment to all the attendees. Each receives the e-mail, drops the iCalendar or Xml attachment onto their particular scheduling application, then their calendars will be updated automatically. Unfortunately if all the time was filtered out due to busy status after organiser completed the scheduling job, scheduling a meeting is unsuccessful, the meeting has to be delayed.

Chapter 3

Analysis and Design

3.1 Possible solution

One possible solution to the problem could be as the one described below. The program first reads an iCalendar or XML file, and transfers its data into the system. Then users could view and manage the data in the system, for example, adding, editing, deleting or scheduling data. Finally, the program should be able to export data in the system to an iCalendar or XML file replacing the original file.

3.2 System Requirments

Before starting the actual design of the program, the targeted platform and required software packages should be considered first. In this project, the program will be coded using c++ language, so Redhat Linux operating system is chosen as the targeted one. Moreover, web-based interface could be used to display the calendar information, therefore software package like ‘Mozilla’ web browser must be required and properly installed in the operating system. In the process of developing the program, more software packages may be required, such as Flex.

3.3 Analyses and Design Diagram

3.3.1 Disadvantages of Similar Calendar Systems

Ximian Evolution Calendar and Microsoft Outlook are well-designed, and used in different system environments. They have a common disadvantage, which is not supporting the import or export of an XML format file. Data in different systems and databases may be in an incompatible format, which leads the problem to become more complex; or even can not be solved at all when the systems try to handle data in different format. Therefore, Data incompatibility and problem complexity can be reduced if the system has the ability to write data to a XML file and convert XML to iCalendar.

3.3.2 C\C++ and Java

C++ is an object-oriented language and a direct descendant of language C that retains almost all features of C as a subset. C++ also provides stronger type checking and supports a wider range of programming styles than C without losing the efficiency.

Java is also an object-oriented language; one important advantage of Java is its portability. A Java program is compiled into byte code rather than a native machine code, byte code can be interpreted by a Java Virtual Machine. If there is a Virtual Machine designed for specific computer architecture, the computer could execute the Java program of byte code directly without compiling the program again. But this feature also causes one of Java's biggest disadvantages. An intermediate path needed first is that Java binary code has to be translated into the equivalent microprocessor instructions by the Java Virtual Machine, but native machine code is a set of instructions that can be accepted directly. Obviously, the binary code translation takes more time on translation, so the performance of Java binary code is slower than that of compiled C++ machine code. In addition, integer value or floating-point value computation in C++ is also several times faster than Java. What's more, C++ handles file I/O much more efficiently than Java does.

In summary, Java performs many times slower than C++, especially in I/O operations. The performance difference between Java and C++ cannot be ignored for my project. One reason is that when my program runs each time, the data in the calendar file will be read in, and additional information also will be written back to the calendar file when the user creates something new or modifies something. Another reason is that there will be a lot of calculations and comparisons in my program. Input file and output file operation is significant in my program. So in order to save user's time I decide to choose the C++ rather than Java for my program. **[1]**

3.3.3 Class Analysis and Diagram

As C++ is an object-oriented language, it was also necessary to draw up a class diagram in order to aid the design process. A class diagram illustrates how the different classes within a program interrelate with each other and also aids the task of programming the code more efficiently. The diagrams for each class and the relationships between classes are shown below to give the reader an idea of the program's overall structure. I have decomposed the system into 5 main classes and one interface main program. They are specified in table 3.3.3.

Time	Handle the time
Date	Handle the date
Datetime	Handle the date and time together
CalendarComponent	Define the calendar components e.g. event, todo
AlarmComponent	Define the alarm part for each calendar component
Calendar	Define some fundamental methods for the calendar system
Interface	Main program, user can run complied program to start to use

Table 3.3.3 Functions of Classes and Main Interface

As can be seen in table 3.3.3, there are three classes used to handle date and time, which are named "`Time`", "`Date`" and "`DateTime`". Class `Time` is to represent the time of day, and class Date is to represent the date. In the calendar system, whenever an event is added or modified, the time and date will be recorded and saved to a file as a single stream. Date will always be written together with time as a property rather than be written separately in different lines as two properties. Therefore the class `DateTime` is used to put the time and the date together as a single object. For example, the created time of an event written in a iCalendar file is "TIMESTAMP:dd/mm/yyyyThh/mm/ss". So the `DateTime` Class make the users control the date and time much more easily and conveniently and clarify the program code. The following figure 3.3.3 shows the relationship among these three classes.

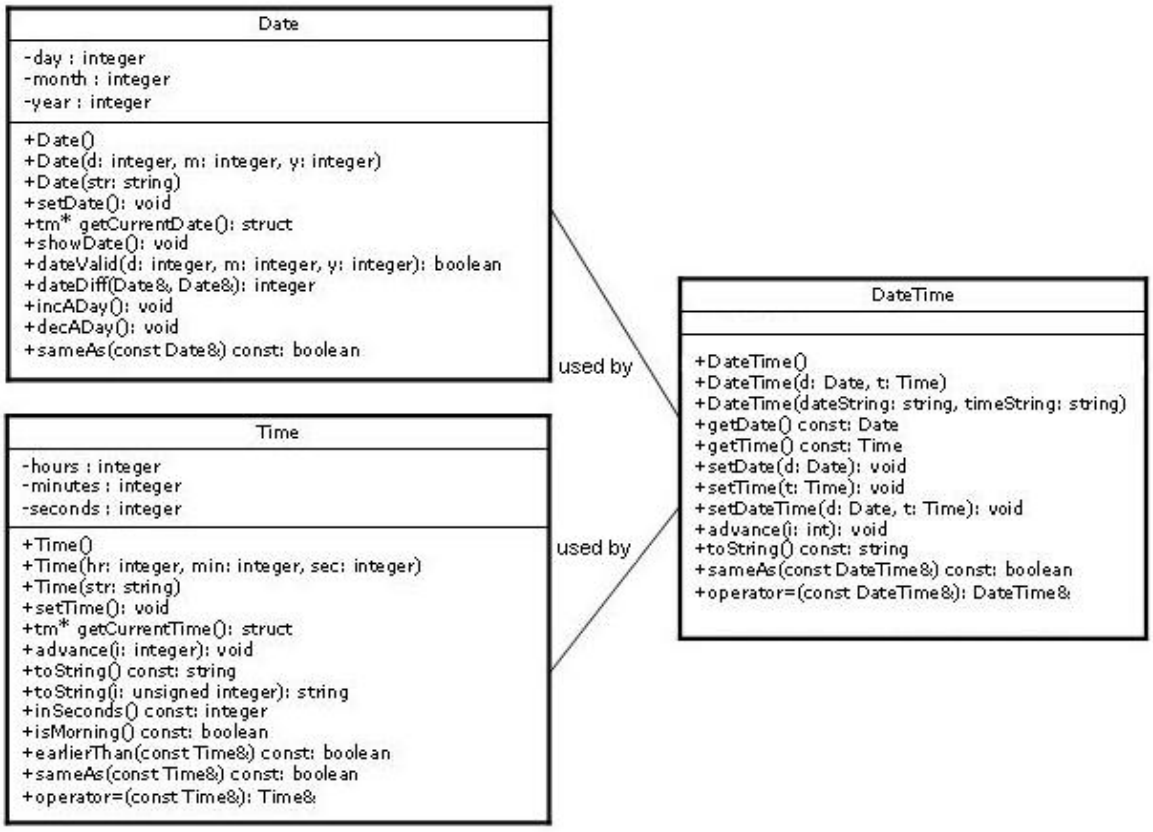

Figure 3.3.3 Date, Time and DateTime Class Diagram

In Figure 3.3.4, There is a class named "`CalendarComponent`", which is a superclass storing the similarities for its subclasses. The `EventComponent` class inherits from the `CalendarComponent` class; and all protected attributes and public operations in the `CalendarComponent` class can be used directly in the `EventCompoent` class. `EventComponent` class may have its own additional attributes and operations. For instance, the transparency attribute of an event only belongs to `EventComponent` class, not to all other subclasses. Each event is an `EventComponent` object.

In addition, the User may add an alarm for an event; there is an attribute in `EventComponent` class with type of `AlarmComponent.` So `AlarmComponent` class is defined here as part of `EventComponent` class.

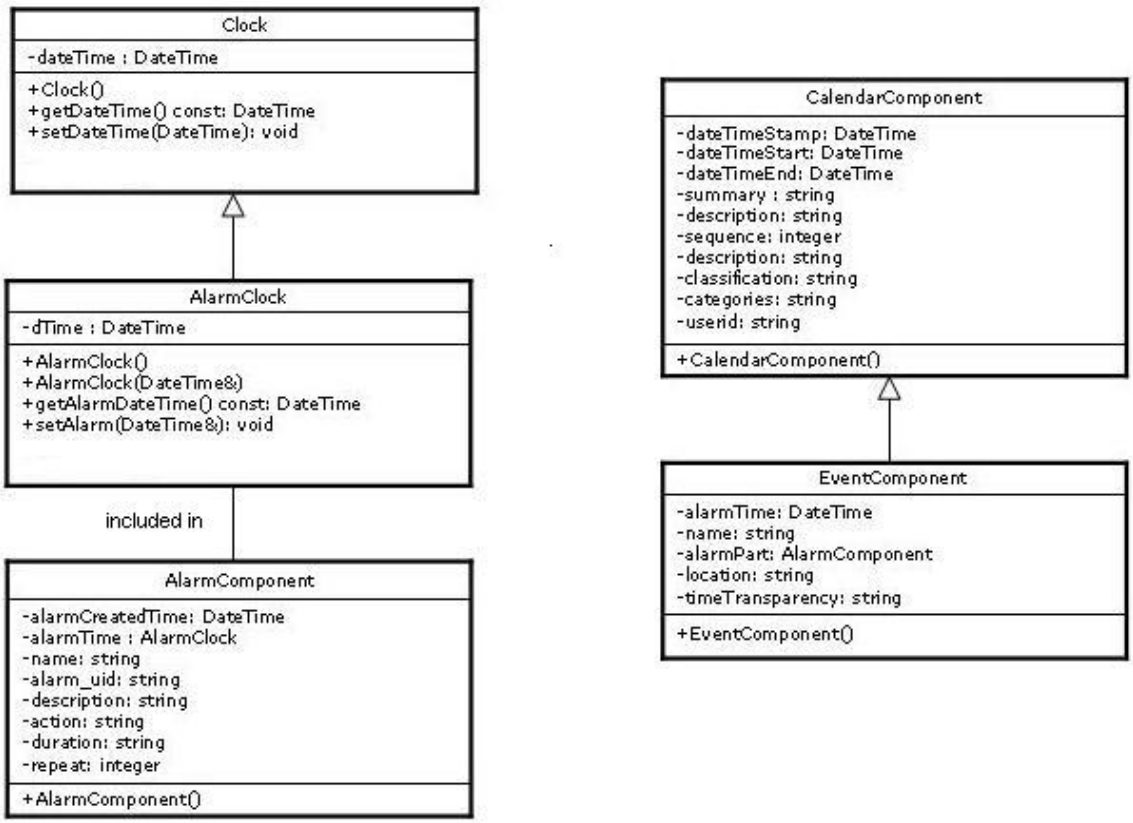

Figure 3.3.4 Calendar Components Diagrams

`Calendar` class is a very important class shown in figure 3.3.5; most of operations will be directly used by the interface of the main program. The table 3.3.5 below shows what the operations do in `Calendar` class.

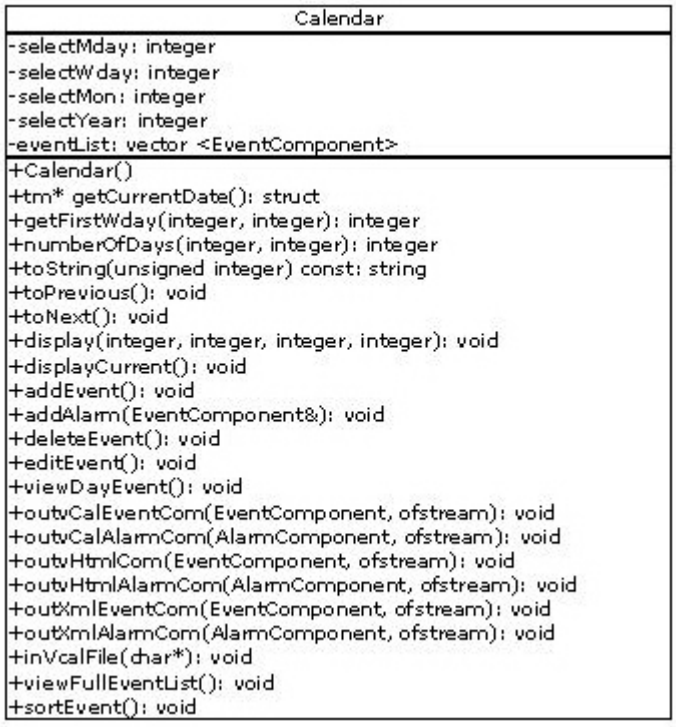

Figure 3.3.5 Calendar Class Diagram

getCurrentDate()	Get the current system time
getFirstWday()	Return the date of the first day in a specific week
numberOfDay()	Count the number of days in a specific month
toString()	Convert an integer to a string
toPrevious()	Return the previous month parameters
toNext()	Return the next month parameters
display()	Display the calendar on screen
displayCurrent()	Display the current month calendar on screen
addEvent()	Add a new event
addAlarm()	Add an alarm for a event
deleteEvent()	Delete an existing event
editEvent()	Edit an existing event
sortEvent()	Sort the events according to the start time of each event
viewDayEvent()	Display the event(s) started today
viewFullEventList()	Display all events in the calendar
outvCalAlarmCom()	Ouput the alarm part to an iCalendar file
outvXmlAlarmCom()	Output the alarm part to a XML file
outvHtmlAlarmCom()	Output the alarm part to a Html file
outvCalEventCom()	Output event information to an iCalendar file
outvXmlEventCom()	Output event information to a XML file
outHtmlCom()	Ouput event information to a HTML file

Table 3.3.5 Meaning of Each Operation in Calendar Class

Interface is an interface program; it only contains a list of public operations that can be implemented by either other classes, or by itself and has no attributes. What's more, Interface provides a way to access the elements of each class, which lead all classes to work as a whole system. Table 3.3.6 will show what the operations do in interface program.

outvCalHeader()	Output the header of iCalendar file
outXmlHeader()	Output the header of XML file
outHtmlHeader()	Output the header of HTML file
fileExist()	Return true if a file exist
isDirectory()	Return true if given name is a directory
checking()	Check the import file is in iCalendar format or XML format
checkingS()	Check the schedule files is in iCalendar or XML format
schedule()	Schedule a meeting for a group of prople
getWday()	Return the first week day of given date

Table 3.3.6 Meaning of Each Operation in Interface program

Attributes and operations for each class have been introduced in the figures above; a class diagram in figure 3.3.6 shows relationships between these classes and how they work together.

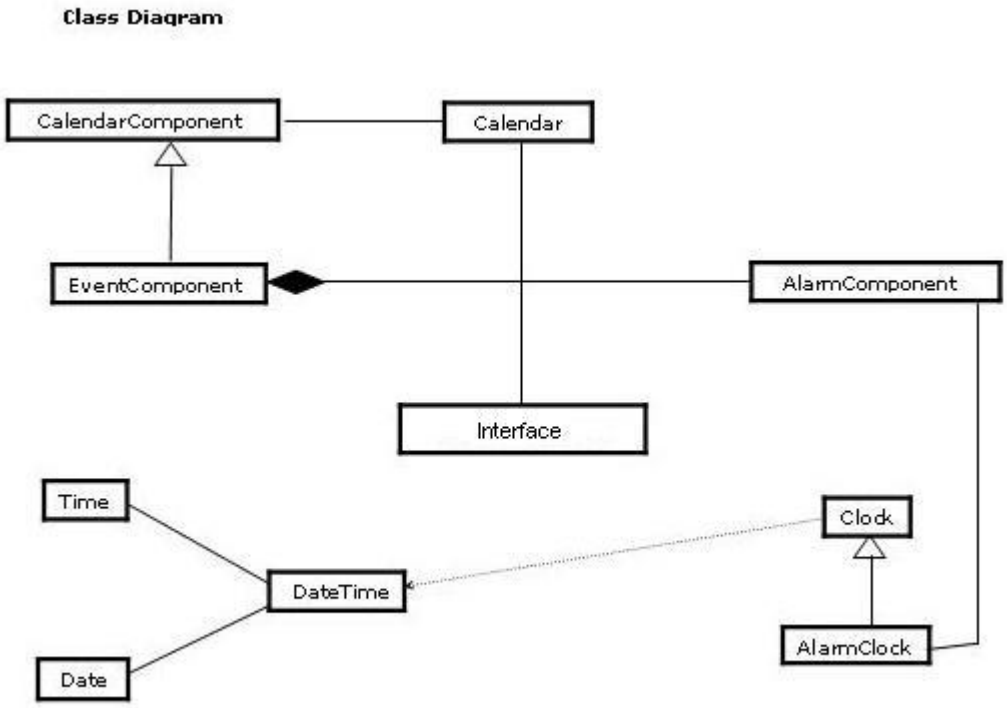

Figure 3.3.6 Class Diagram

3.4 Analyse the iCalendar Data Structure

In a iCalendar file, the string “BEGIN:VCALENDAR” appears as the first line, and the string “END:ICALENDAR” appears as the last line, all data between these two lines will be extracted and stored in a vector when the program is run. The number of entities in a file determines the size of the vector. If there are three event entities in file, when data was read in by the program, the size of the vector would be three. Each time a user adds a new event; the size of the vector will increase by 1.

When a user closes the current program, all data will be written back to the iCalendar file. Firstly, the string “BEGIN:VCALENDAR” will be written as the first line in an iCalendar file. Secondly, the information about each event will be written back to an iCalendar file beginning with string “BEGIN:VEVENT” and terminating with the string “END:VEVENT”. If the information about summary, description, category and location for an event was not input by user, the corresponding field of the event would be empty and not be written in the iCalendar file. Finally when all data is written back to the iCalendar file completely, the string “END:VCALENDAR” will be written as last line to close the iCalendar object, and the file exporting is finished. Altogether the iCalendar date

structure has been constructed completely. So the program can always import a right iCalendar file and export calendar data to an iCalendar file with correct data structure. **[12]**

3.5 XML Lexical Analysis

Calendar information will also be saved in an xml format file. The program should have the ability to read data from an xml format file, because the organiser may receive xml format calendar files from attendees rather than iCalendar format files when he/she is going to schedule a meeting. The program will convert an xml format file into an iCalendar format file using a lexical analyser first, and then by reading data from that generated iCalendar format file when the user supplies an xml format file to the program. The generated iCalendar format file can be converted from original xml format file without loss of any calendaring information. For example, if an xml format file called "`calendar.xml`" is supplied to the program, the program will generate the `calendar.xml.ics` automatically - and then import the data from the `calendar.xml.ics` file. So the program will succeed in handling the xml format file.

3.6 Analysis Scheduling Problem

Scheduling a meeting for a group of people is an important functionality and feature in a calendaring and scheduling application. My calendar program cannot connect to an exchange server, so the calendar program will need to get all scheduling files in order to start a new scheduling job. One thing that needs to be considered is that all received scheduling files should be accepted by the calendar program in either iCalendar file format or xml file format because the people an organiser wants to invite may use a variety of scheduling applications or different operating systems. What's more, people may only want to work from 9am to 5pm in a working day and do not want to work at the weekend, so the system should supply some options to the user - who can choose to schedule a meeting for the working time or for 24 hours of the day. The scheduling function will reduce the amount of work, improve the efficiency of work; save the user' time. Assumption can be made here that an event must at least last thirty minutes, so there should be 48 thirty minutes in a day. So I decide to declare one 48*7 array of integer type, which is used to store the free and busy time for a week for a group of people. All entries of this array could be initialised with value 0s. When an attendee is busy at a time on a day, value in the entry associated with that date time will be changed to 1. The program did not finish until all of events had been scheduled. The time complexity of this schedule algorithm can be considered as following:

The best case

There are n events in total from attendees' scheduling files, each event lasts just 30 minutes. The scheduling function needs several comparisons for each number of n and needs n assignments totally which are changing the values of related entries to 1. The comparisons are varying depending on different cases. Therefore, the scheduling function can be completed in linear time. The time complexity of the scheduling function is O(n).

The worse case

There are n events in total from attendees' scheduling files, and all of events will last for 7 days, which the users specified. The scheduling function also needs several comparisons and 7*48 = 336 assignments for each number of n. Therefore, the scheduling algorithm needs at least 336n times to complete. If the size of the problem n is much and much greater than 336, the time complexity is still O(n). if the size of the problem is much less than 336, the time complexity of the scheduling function may be $O(n^2)$, $O(n^3)$ or even much bigger.

In sum, the scheduling function is an efficient algorithm for large size of the problem, but it may not be the best choice for small size of problem in some cases.

3.7 User Interface Design

"The term user interface refers to the methods and devices that are used to accommodate interaction between machines and the human beings who use them. There are two fundamental tasks: communicating information from machine to the user, and communicating information from the user to machine". **[14]**

Graphic User interface design is a very important job for widely acceptable software; software with well-designed interface will make it easier and more interesting to use for people. The user interface is also quite important and cannot be ignored in this project. Let us consider what will happen if the users interface design is ignored here. Firstly, the required information and required results should be displayed following the prompt; a user may not recognise which part of the information is wanted and which part of the information is system command. Secondly, part of the information will be rolled up when the amount of information needed to be displayed is overloaded - which always leads a user to make mistakes.

A web-based interface design is a common and effective method, so I was going to display the user required information on a web browser, such as full event list, user manual, scheduling results. For an instance, when a user is choose to view details of events, scheduling results or the user manual, the system will start up a web browser automatically with the user expectative information. Of course, the large amount of information displayed is not a problem any more, the reason is the users have been used to using a scroll bar to view the information on a web browser.

Chapter 4

Implementation Experience

4.1 Introduction

Some functions had been built and worked correctly in the final completed product. Once the program is run, the current month's calendar will be displayed on screen, and the user can choose the jobs to do from the system main menu. These jobs are listed below:

1. View Previous Month Calendar
2. View Next Month Calendar
3. View Today's Event(s)
4. View Full Event List
5. Add New Event
6. Edit Event
7. Delete an Existing Event
8. Schedule Meeting For a Group
9. View the User Manual

Later sections of this chapter will tell readers how these functions were implemented, and what problems had been met during coding the function, and what methodologies had been used to solve the problems.

4.2 Fundamental Functions Implementation

4.2.1 View Previous and Next Month Calendar

Each time the program is started, the current month's calendar will be displayed on the screen, and the user will know the current date and time at once. The display method with four parameters; which are month, year, month day, week day, was defined to do this job. In order to display the current month's calendar, the current system date and time is caught and passed to the display method each time.

In the `display` method, in order to display the date in the correct format, like the format in figure 4.1, thus we use week day of today and the number of days in the current month to calculate the week day of 1st of current month, for example, the 1st day of April in 2003 is Tuesday, so number 1 is located at the "Tuesday" column, and other numbers follow. So the program will always show the user the correct well-formatted calendar information unless the system time is incorrect. **[11]**

April 2003

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

Figure 4.1 An Example Of Calendar Layout

A user can make a choice to view a previous month's calendar, so the `toPrevious` method was used to change the values of the current time received from the system, and then passed the modified values to the `display` method such that the previous month's calendar will be displayed. In the `toPrevious` method, if the value of the current month is 1, then the value of the last month must be 12 and the value of the year will be decreased 1. If value of current month is between 2 and 12, then value of last month is value of current month minus 1 and value of year does not change. Values of the other two parameters also can be deduced from the current system time. `toNext` method in the program was used to change values of current system time to values associated with next month; `display` method with values modified by `toNext` method will show next month's calendar information to the user. Implementation of `toNext` method is similar with the implementation of `toPrevious` method.

4.2.2 View Full Event List or Today's Event(s)

When the program is executed, all events are saved in a vector, and users can make a choice to view the full event list. The size of the vector determines the number of events stored in a user's calendar, so we use a "for" loop to output details of all events on screen, with indices. If users choose to view their today's event(s), the start time of each event must be compared with the date for today in the "for" loop; if the start time of an event is equal to today's date, then the details of the event will be printed on screen. Otherwise, all the other days' events will be ignored. The purpose is that users are reminded what they should do today according to the event list that is received.

4.2.3 Add A New Event

Events are personal items that only users are required to attend. Creating an event in your calendar reminds you to plan around that time. Examples of events include a visit with a doctor, picking up a child from school, or a due date on a project. Each time when the users choose to add a new event into their calendar, the program will ask users to input text from the keyboard, and input values will be assigned to the corresponding part of an event, and push the new event to the end of the vector.

In the `addEvent` method, the default duration of an event was half an hour - if the user did not input a start time and end time of a new event, then the start time of the event would be set to the current system time, and the end time of the event would be 30 minutes later. Another point I noticed was that the user may input several words, sentences and even more than one paragraph for a description and the summary fields of event, but `cin` can only read the input string without white space between it, so I was going to use `get` method or `getline` method to deal with description and summary input. `Get` method and `getline` method both take three arguments, the last argument is the terminating character. `Get` method and `getline` method will stop reading input when a terminating character is read. A terminating character has the default value of '\n', therefore both `get` method and `getline` method store input until the users press carriage return. But the `get` method does not extract the delimiter from the input stream when it stops. On the contrast, `getline` method can work without this problem. Thus I decided to use `getline` method in my `addEvent` method. **[6]**

Unfortunately, I met another problem when I had finished the `addEvent` method, the program can be compiled, but the program did not return an input prompt, and it did not allow the user to input text anywhere. Eventually, I discovered every `getline` method function did not work

correctly after a `cin`. The reason is `cin` has a similar problem with `get` method, it has a default terminating character '\n' and can not extract '\n' from the input stream, so I used a piece of code `getline (cin, consume)` to consume '\n' of `cin` before using `getline` method. In the result, the `addEvent` method worked properly.

4.2.4 Edit and Delete An Event

Users can edit an existing event when the information of an event has been changed. When users select edit event, all events will be listed with an index, and the users are required to enter the index number they want to edit. After the index number was input, details of the required event and a sub menu would be displayed on the screen at the same time, users can make the choice to edit any field of the information of the event, and then write back the vector. Thus editing an event is completed.

In addition, if an event was past or cancelled, users can delete it from their calendars. When users select delete event, users also need to enter an index number they want to delete. If the index number equals to the size of vector, that means the user expects to delete the last event in the vector, so we can simply resize the vector by reducing the size of the vector by 1, the last element of the vector will be lost automatically. Otherwise, all elements with a higher index have to been moved up one place; then decrease the size of the vector by 1. **[7]** Whenever users delete an event, the vector needs to be resized in order to improve the efficiency and the correctness of other methods. For instance, the resize vector will reduce the number of iterations basically in the `viewFullEventList` and `viewDayEvent` methods. (Source codes are available in Appendix C)

4.2.5 See the User Manual

The user manual, which is essential part for a software product, is a guide and shows the people what and how the system does step by step. People can read through the user manual in order to grasp how to work with the system quickly.

A user manual file named "`user_manual.html`" is containing the instruction which tell the user how to use the current system. When the user chooses to view the user manual, the several system commands will be executed automatically to do the following tow jobs. One is that the system will copy the "`user_manual.html`" file to `/tmp/` directory, and then open a web browser to display the information in that file. Moreover, there is a list on the top of the displayed web page, the users can click the topic to reach the expected information directly.

4.3 Calendar File Handling

In this project, the program support import of either iCalendar file or Xml file, and has the ability to export calendaring information into files in both formats. All calendaring information will be output to an iCalendar file named `calendar.ics` and an xml file named `calendar.xml` when the program is closed each time. As a well-formatted iCalendar file, string “BEGIN:VCALENDAR” must be written into the `calendar.ics` file as the first line; then write calendaring information back to the file, each event entity is also beginning with line “BEGIN:VEVENT” and closing with line “END:VEVENT”. Finally the line “END:VCALENDAR” must be written to close the `calendar.ics` file. In addition, there are no white spaces at the start of each line in the `calendar.ics` file. As a well-formatted Xml file, every element in the file must have a closing tag and the white spaces are used to represent the relationships between elements. (Sample output files are available in Appendix D)

iCalendar data structure has been mentioned in section 2.1 and section 3.2. Xml syntax and data structure has been mentioned in section 2.2. So this section will focus on how the program will check whether files exist, and how the program imports a file if it exists. In this project, a command line argument is used to allow the user to supply a file name to program. If the user did not supply a file, the program would import `calendar.ics` by default, otherwise the program would import the file supplied. What’s more, the file supplied must exist, or the warning message “Unable to open the file” will be displayed and the program exits.

4.3.1 Check Whether a File Exists

I had even compared three different methods to check if a file exists. These methods are shown in table 4.3.1 below:

`<1> ifstream FileObject("file name");` `if (FileObject.fail()) { //file does not exist.}`
`<2> bool FileExistViaAccess(const char* FileName) {` `return (access(FileName, 0) == 0);}`

`<3> bool fileExist(const char* fileName) {`
`struct stat my_stat;`
`return (stat(filename, &my_stat) == 0);}`

Table 4.3.1 Methods for File Existing

Method <1> would only work if the file was in the same directory as the program, but the file would be stored in different directory as the program in the real case. Method <2> tries to access the given file in order to determine whether a file exists or not, so this method is not efficient. Method <3> is using stat to determine if a file exists, and also the method <3> using stat can be passed a path as an argument. So method <3> is more practical and the most portable to make that determination. Method <3> can be a success to determine if a file exists, but it is also important to know if the file is a directory or a file, because a file cannot be created if its name is the same as a directory in the same directory. Therefore I use the `isDirectory` method to cooperate with method <3> in the program. ***[5]***

4.3.2 Import An Existing File

iCalendar file has a fixed format, the first several characters of each line will determine what information is followed. For instance, “CLASS:public” represents a public event, so when the program reads the line “CLASS:public”, classification of the corresponding event would be set the value as public. Most lines in the iCalendar file can be handled in similar way. When the program reads “END:VEVENT”, it would push the event to the vector and start to read the next event; finally all events would be imported successfully.

Xml file also has a fixed format, but each element in the file has a closed tag. So the above method will only work for the iCalendar file. Therefore, I decided to use flex to extract the information between tags and then written them to a new file with iCalendar format, and then apply the above method to read the calendaring information from that new iCalendar file into the program. Flex has the ability to activate rules conditionally; any rule whose pattern is prefixed with <cond> will be active only when analyser is in start condition cond. These start conditions need to be declared in the definition section using %s in the flex file. Each start condition cond is activated using BEGIN(cond) action, and this condition applies until next BEING action is executed.

Here is an example that will give a reader some ideas about how flex works. When flex is dealing with the line <item>Education</item>, first flex checks the line and the start condition is activated using BEGIN(CAT) action, then extract data between <item> and </item> into a buffer, and then write the date in the buffer to a iCalendar file as CATEGORIES:Education, finally flex clean the buffer by using BEGIN(INITIAL) action and continue to deal with the next line until reaching the last line. Lexical analyser file named xml.lex can be found in appendix C.

4.4 Scheduling Implementation

The schedule method is divided into two parts, one is scheduling attendees' calendar files, and another is displaying the scheduling result with the expected format. Firstly, I will introduce how to schedule a group of calendar files. Assume every event will last at least 30 minutes. I declared two 48*7 arrays, where 48 represents the fact that there are 48 half hours in a day and 7 represents the fact that there are 7 days a week. One array called "status" only store the digit 0 or 1, another array called "display" stores the "Free" or Busy" values. The meeting organiser will be asked to give a date in order that the program works out result for the week including the given date. The value of each entry in the display array is "Busy" and in the status array is 0 originally. During scheduling, if any attendee is busy at a time of that week, the value of the corresponding time entry in the status array will be changed to 1 until the schedule is finished. And then the value of the entry in the display array will be changed to "Free" if the value of corresponding entry in the status array is still 0. So the organiser can arrange a meeting at the time, with a value "Free", otherwise they can try to schedule a meeting with other specified time.

Displaying the scheduling result will depend on the organiser's choice, if the organiser chooses to schedule only for working time, part of the scheduling result between 9am and 5pm will be displayed. If the organiser chooses to schedule for all of the time, the 24 hours full scheduling result will be displayed.

4.5 Interface Implementation

A user can view all events and scheduling results on the Mozilla web browser. Like calendar.ics and calendar.xml, the program will also create an html file named calendar.html in a shard folder to store details of all events. Each time when the user chooses to view all events, the system will run Mozilla automatically, and all events information will be displayed and well formatted on it.

After each scheduling job is completed, the scheduling result also would be formatted and saved in schedule.html, and the system will run Mozilla to open that schedule.html automatically. In addition, the scheduling result will be displayed in different colour in the web browser to reduce user's effort.

4.6 Other Techniques Used

4.6.1 Ostringstream Class

There is a method `toString` in `Time, Date`, and `DateTime` classes respectively, which is used to convert the Time, Date or DateTime object to a string through using `ostringstream`

class. The `ostringstream` class provides a buffered stream, and returns a string containing the information that is extracted from stream by using `str` method. ***[15]*** Note that ends must be added at the end of stream instead of `endl,` because `ends` inserts the null character "\0" in the `ostringstream`, and `endl` adds a new line at the end of stream and flushes the stream. In addition, remembering to include a new library `<sstream>` to instead that outdated and deprecated header `<strstream>,` and use `ostringstream` rather than `ostrstream` to avoid warning message. Altogether, `ostringstream` is really easy to use; any types can be passed into it so long as a stream handler exists for it***. [16]***

4.6.2 Get System Time

The current time needs be retrieved from system and passed to a date or time object. In order to get current system time, a timer with type of time_t is declared, which can get current calendar time in seconds by passing a "NULL" to time funtion, and then apply localtime function to convert the timer from type time_t to struct tm that holds the current date and time in usable form. The struct tm has structure shown as below:

```
struct tm {

   int tm_sec;  // second (0-59)
   int tm_min;  // minute (0-59)
   int tm_hour; // hour(0-23)
   int tm_mday; // day (1-31)
   int tm_mon;  // month (0-11)
   int tm_year; // year
   int tm_wday; // weekday (0-6), Sunday as 0
   int tm_yday; // yearday (0-365)
   int tm_isdst;// Nonzero if daylight saving time is in effect.
};
```

Getting the current system time is achieved and each arributes in the struct tm can be called and used directly. ***[17]***

Chapter 5

Evaluation

5.1 Introduction

In order to effectively evaluate the solution produced by this project it is necessary to examine it in terms of how well it have fulfilled the initial objectives and how well these objectives fulfil the problem needs depending on users' feedback.

5.2 Fulfilment of Initial Objectives

The project can be seen have successfully delivered all initial objectives in the project life cycle. The system successfully implements the aim of viewing and managing people's appointments. The objectives of this project were also extended, because scheduling a meeting successfully delivered its target of calculating the most convenient result to the organiser with the minimum of effort.

5.3 Advantages of Final Product by Comparing With Other Products

I have mentioned similar software products, such as Ximian Evolution Calendar, MSN Calendar, in earlier chapter. Both of them can store and exchange calendaring and scheduling information through using the file with iCalendar format rather than XML format. By comparing the current system with them, the current system can not only deal with the iCalendar format, but also handle the XML iCalendar documentation (xCal) by converting the XML format to the standard iCalendar format. In

addition, the XML is gaining widespread attention as a 'web friendly' syntax for representing and exchanging documents and data on the Internet. Therefore, one advantage is that the XML file generated by the system can be exchanged between incompatible systems and accepted by many more applications.

5.4 User Feedback

Final testing with users to determine the usability of the system shows that although the initial objectives of the project have been successfully met there is still room for improvement in the system.

People with little Knowledge of Computing

The user pointed out the difficulties involved in adding a new event into the calendar as there are too many information required to enter using keyboard by user. In addition, they mentioned the data had to been typed in the correct format, otherwise the system will quit suddenly. For example, when the user input unmatched formatted information for EVENT START TIME, the system will quit. All of these are limiting the efficiency of the calendar tool currently. The project had aimed to make this operation as simple as possible so as to make the system usable by people with little knowledge of computing. This again indicated a possible future modification whereby the adding event function could provide a friendly graphic user interface (GUI) instead of the current text based interface in order to reduce the user's effort.

People with Computer skills

It was noted by test subjects that viewing previous and next month's calendar functions are quite convenient to use, but the method to view any month calendar has not been built in the system currently. However some other calendar products support this operation. So the calendar system can be developed by providing go to any month calendar method in order to supply user more choice. Therefore, people can view any month calendar directly by giving a month and a year values.

The users also points out the difficulties involved in scheduling a meeting function as user has to type all attendees' calendar files into system that wastes user's time and reduces the correctness of the system. A possible future modification of scheduling a meeting function is to only require user to input the name of a folder which store all attendees' calendar files, and then all of free/busy information of calendar files in that folder will be read into the system automatically.

5.5 Testing

Testing is an efficient way to detect the bugs existing in the program. Each class was tested fully when completed; further testing was done when the final program was finished. The following show the tested results, what bugs had been detected and how to fix these bugs for some functions.

Test 1. Make a Choice form System Menu

When the final program was just completed, the user only was allowed to make a choice by inputting a lowercase letter from the menu, therefore when the user input a capital letter or any other invalid string, the system would do nothing which caused the confusion. There were 4 menus in total in the program; all of them had this bug mention above. Therefore the program was modified which treated both lowercase and capital letter as valid input. When the user input an invalid string, the system will output error message and ask the user to input choice again. The confusion involved earlier was solved, and the system will not abort any more.

Test 2 View Full Events Information on Web Browser

There was a problem that had been encountered during testing the View Full Events function. The events that the user just added could not be displayed on the web browser. The reason was that the program would only write the calendaring information back to the iCalendar, XML and HTML file when the user chose to quit the system. When the user was going to view the full events list, the web browser would open the old HTML file without including the new added data. So the information displayed on web browser was incorrect.

One approach to view correct information was that the user could simply restart the program and chose view the full event list option again, the information would be displayed as expected, but this approach is meaningless and can not be accepted. Another approach was to modify the program which would output all calendar data to ‘calendar.html’ firstly, then run web browser to open ‘calendar.html’ so long as the user chose view full events list option. The second approach is the best way which solves the problem completely.

How the events information displayed on web browser is shown below in figure 5.5.1

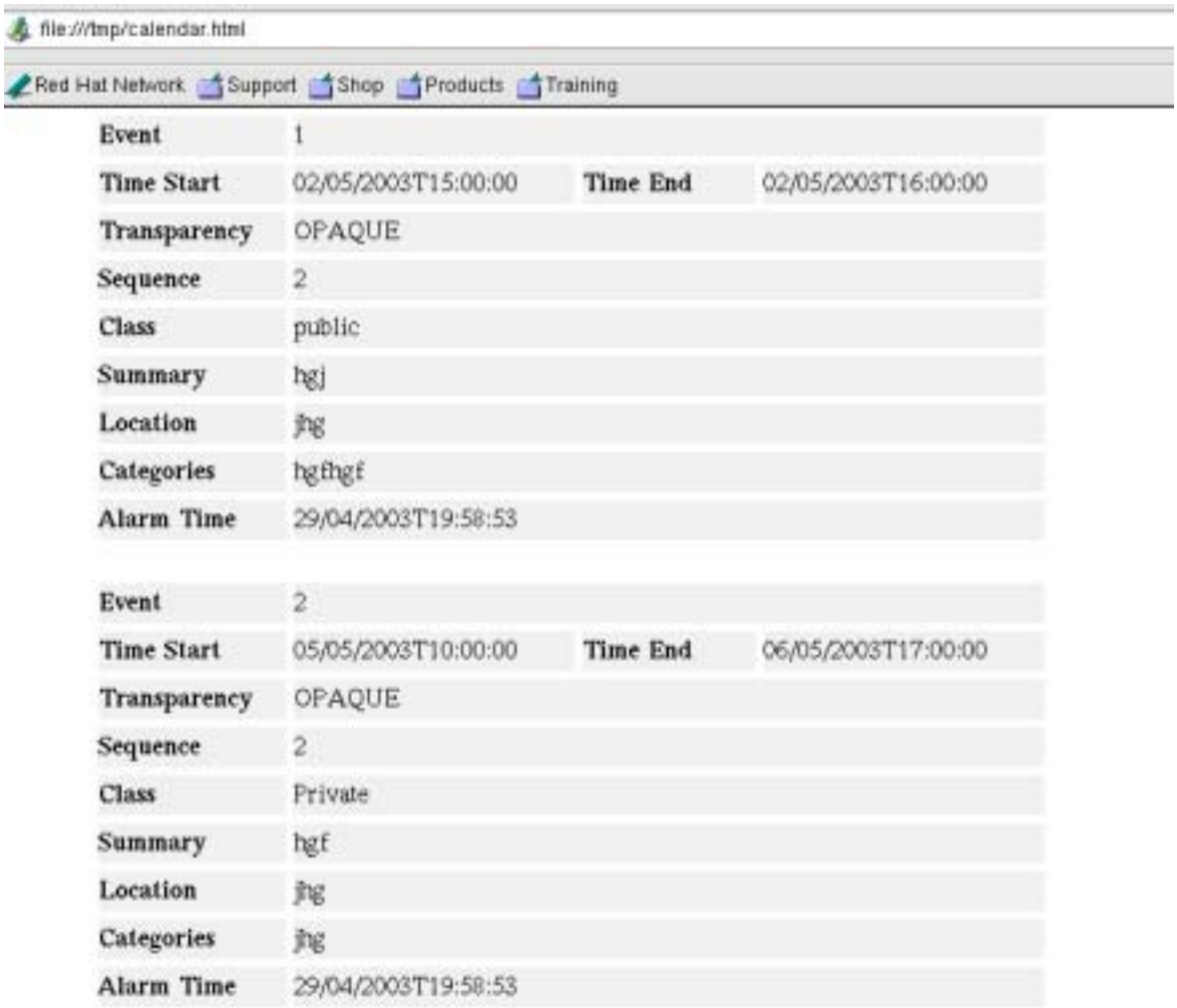

Figure 5.5.1 Displaying the Event Information on Web

Test 3 Scheduling Function

Scheduling Function was tested fully, and the scheduling results were correct displayed on web browser for any number of the scheduling files. The following two figures show the scheduling results which calculated from scheduling the single scheduling file “icalendar.ics” for the week starting from 13/04/2003 to 19/04/2003. There was only one event that took place this week on 14/04/2003, and this event also was an all day event. The scheduling results were calculated correctly and displayed on web browser as expected. (See appendix D)

	Sun 13/04/2003	Mon 14/04/2003	Tue 15/04/2003	Wed 16/04/2003	Thu 17/04/2003	Fri 18/04/2003	Sat 19/04/2003
9:0	Free	Busy	Free	Free	Free	Free	Free
9:30	Free	Busy	Free	Free	Free	Free	Free
10:0	Free	Busy	Free	Free	Free	Free	Free
10:30	Free	Busy	Free	Free	Free	Free	Free
11:0	Free	Busy	Free	Free	Free	Free	Free
11:30	Free	Busy	Free	Free	Free	Free	Free
12:0	Free	Busy	Free	Free	Free	Free	Free
12:30	Free	Busy	Free	Free	Free	Free	Free
13:0	Free	Busy	Free	Free	Free	Free	Free
13:30	Free	Busy	Free	Free	Free	Free	Free
14:0	Free	Busy	Free	Free	Free	Free	Free
14:30	Free	Busy	Free	Free	Free	Free	Free
15:0	Free	Busy	Free	Free	Free	Free	Free
15:30	Free	Busy	Free	Free	Free	Free	Free
16:0	Free	Busy	Free	Free	Free	Free	Free
16:30	Free	Busy	Free	Free	Free	Free	Free

Figure 5.5.2 Scheduling Result for Working Time

	Sun 13/04/2003	Mon 14/04/2003	Tue 15/04/2003	Wed 16/04/2003	Thu 17/04/2003	Fri 18/04/2003	Sat 19/04/2003
0:0	Free	Busy	Free	Free	Free	Free	Free
0:30	Free	Busy	Free	Free	Free	Free	Free
1:0	Free	Busy	Free	Free	Free	Free	Free
1:30	Free	Busy	Free	Free	Free	Free	Free
2:0	Free	Busy	Free	Free	Free	Free	Free
2:30	Free	Busy	Free	Free	Free	Free	Free
3:0	Free	Busy	Free	Free	Free	Free	Free
3:30	Free	Busy	Free	Free	Free	Free	Free
4:0	Free	Busy	Free	Free	Free	Free	Free
4:30	Free	Busy	Free	Free	Free	Free	Free
5:0	Free	Busy	Free	Free	Free	Free	Free
5:30	Free	Busy	Free	Free	Free	Free	Free
6:0	Free	Busy	Free	Free	Free	Free	Free
6:30	Free	Busy	Free	Free	Free	Free	Free
7:0	Free	Busy	Free	Free	Free	Free	Free
7:30	Free	Busy	Free	Free	Free	Free	Free
8:0	Free	Busy	Free	Free	Free	Free	Free
8:30	Free	Busy	Free	Free	Free	Free	Free
9:0	Free	Busy	Free	Free	Free	Free	Free
9:30	Free	Busy	Free	Free	Free	Free	Free
10:0	Free	Busy	Free	Free	Free	Free	Free
10:30	Free	Busy	Free	Free	Free	Free	Free
11:0	Free	Busy	Free	Free	Free	Free	Free
11:30	Free	Busy	Free	Free	Free	Free	Free
12:0	Free	Busy	Free	Free	Free	Free	Free
12:30	Free	Busy	Free	Free	Free	Free	Free
13:0	Free	Busy	Free	Free	Free	Free	Free
13:30	Free	Busy	Free	Free	Free	Free	Free
14:0	Free	Busy	Free	Free	Free	Free	Free
14:30	Free	Busy	Free	Free	Free	Free	Free
15:0	Free	Busy	Free	Free	Free	Free	Free
15:30	Free	Busy	Free	Free	Free	Free	Free
16:0	Free	Busy	Free	Free	Free	Free	Free
16:30	Free	Busy	Free	Free	Free	Free	Free
17:0	Free	Busy	Free	Free	Free	Free	Free
17:30	Free	Busy	Free	Free	Free	Free	Free
18:0	Free	Busy	Free	Free	Free	Free	Free
18:30	Free	Busy	Free	Free	Free	Free	Free
19:0	Free	Busy	Free	Free	Free	Free	Free
19:30	Free	Busy	Free	Free	Free	Free	Free
20:0	Free	Busy	Free	Free	Free	Free	Free
20:30	Free	Busy	Free	Free	Free	Free	Free
21:0	Free	Busy	Free	Free	Free	Free	Free
21:30	Free	Busy	Free	Free	Free	Free	Free
22:0	Free	Busy	Free	Free	Free	Free	Free
22:30	Free	Busy	Free	Free	Free	Free	Free
23:0	Free	Busy	Free	Free	Free	Free	Free
23:30	Free	Busy	Free	Free	Free	Free	Free

Figure 5.5.3 Scheduling Result for 24 Hours

5.6 User Instruction

5.6.1 Getting Started

1. On Linux systems, copy files from CD to your own directory and compile the main program called Interface.cc using Makefile with the command:

make Interface

and an executable program Interface will be generated, and then type command ./Interface to start calendar.

2. Current month's calendar and calendar system menu are displayed like shown below. You can view current month's calendaring information or enter a choice to start calendar activities.

```
            May 2003
 Sun Mon Tue Wed Thu Fri Sat
                     1   2   3
   4   5   6   7   8   9  10
  11  12  13  14  15  16  17
  18  19  20  21  22  23  24
  25  26  27  28  29  30  31
Calendar System 1.0

Menu List:

     a)  View Previous Month Calendar
     b)  View Next Month Calendar
     c)  View Today's Event(s)
     d)  Add New Event
     e)  Edit An Event
     f)  Delete An Event
     g)  View Full Event List
     s)  Schedule Meeting For A Group
     h) Help (User Manual)
     q)  Quit

Enter your Choice:
```

5.6.2 Previous or Next Month's Calendar

To view previous or next month's calendar by choosing option a) or b) from the calendar system main menu, then previous or next month's calendar will be shown. You can apply the option repeated to view any either future month or past month's calendar you want.

5.6.3 Viewing Events

1. To view today's event by choosing the option c) from the calendar system main menu, some of events that starts today will be displayed.

2. To view full event list by choosing the option g) form the calendar system main menu, all of events will be displayed with an index. You can know how many events in total in your calendar program.

5.6.4 Creating, Deleting or Editing an Event

1. To create a new event by choosing the option d) from the calendar system main menu.

2. Fill in the *Summary* field and enter any other relevant information.

3. If you would like to add a reminder for this event, give an answer "yes" to the question "Do you want to add a reminder for this event", and enter relevant reminder information.

4. To delete an event by choosing the option f) from the calendar system menu. All of events are displayed with an index, and then you can enter an index number to delete it.

5. To edit an event by choosing the option e) from the calendar system menu. All of events are displayed with an index, and then you can enter an index number to edit it. At the time, you can see an edit list sub menu as shown below and choose the option to change any field of the event. The option j) will bring you back to main menu.

Edit List:

a) Change Event Start Time
b) Change Event End Time
c) Change Free or Busy Time
d) Change Classification
e) Change Summary

f) Change Location
g) Change Categories
h) Change Description
i) Change Alarm
j) Finish
Enter your Choice:

5.6.5 Scheduling a Meeting

1. To schedule a meeting by choosing the option s) from the calendar system main menu.

2. A sub menu is shown, you can choose the option a) in sub menu to scheduling a meeting at working time or the option b) to schedule a meeting for 24 hours.

3. Enter a date, number of files and names of files to start then schedule job.

5.6.6 Getting Help and See User Manual

1. To get help by choosing the option h) from the calendar system main menu.

2. A web page will be displayed to show the user instruction.

3. Follow the links at the top of the web page to reach the expectative information directly.

5.6.7 Quit

1. To leave the system by choosing the option q) from the calendar system main menu.

Chapter 6

Further Development

6.1 Introduction

The scale of the project may be further developed. Some ideas for improving the system overall usability and usefulness were uncovered due to the time available for the delivery of the solution. The ideas which could not be developed because of time restrictions are outlined below and serve as an indication of what possible development of the system may achieve.

6.2 Graphic User Interface

The text based user interface is bit confuse sometimes, for example, when the users added new event and scheduled a meeting, they had to follow the system output and type information through keyboard one by one without jumping. This would allow the implementation of flexible graphic user interface, which allows the user to choose the function by clicking mouse on that function button and type the information into text boxes optionally. Thus the confusion currently involved should be eliminated. For example, if the users chose wanted to add a new event, they were at least needed to type text into the summary box rather than fill all fields of the event, then saved that event. The data of all other fields would be passed by the default values.

Therefore the graphic user interface makes the users' lives easier. The percentage of making mistakes will be decreased dynamically. What's more, the colour, pictures, audio and animation can be added into the user interface that makes the software more attractive.

6.3 Multi-user Management

At the moment, the current system only supports the single user. All calendaring information would be written into a file with default file name in default directory. Assume the current system was used by two users; they would be confused by another's calendar data. Therefore, the system could be expanded to support two or more than two users. Each time a user log on the system, the identification number associated with that user will be passed to the system, and then the system will load all calendar data of that user from the calendar file. This would allow the implementation of an access control interface, which allows registered users to log on the system by input user id and password, and unregistered users to create their own new account. Thus the confusion currently involved should be eliminated.

Unfortunately, there was limited time left before the final project deadline and there was doubt as to whether the access control interface could be implemented in time. Hope it can be completed in the near future.

6.4 ToDo Type of Calendaring and Scheduling Entities

In the current system, only event entities are included in the iCalendar object. In other words, the current system only allows adding event information into the calendar system. In fact, the iCalendar can be used for exchanging information not only about event, but also todo types of calendaring and scheduling entities. A todo is a calendaring and scheduling entity that represents an action-item or assignment. For example, it may be an item of work assigned to an individual; such as "John to buy some gifts for the party tonight". This would require the system to be further developed and allow users to add, edit and delete todo information. Obviously, the usability and the compatibility of the system will be enhanced.

6.5 Searching Function

In the current system, there are only two options which can be chosen by users to view either today's event list or full event list. However, the user may expect to view the events according to the category, the summary or the date. The system could be expanded to supply the search function which would

allow the user to view the exact event(s) what they are looking for by entering a few words into associated search field. For example, if a user enters “Education” into the category search field, all events within Education category will be displayed on the screen. One advantage of this search function is that users can narrow the results returned from the current system and find the result in a faster way with the minimum effort.

6.6 iCalendar Limitation

The system currently only supports the single iCalendar object in a single data stream. However, the iCalendar format should have the capability for including multiple individual iCalendar objects in a single data stream according to the iCalendar specification. The system may be possible to be further developed to allow any number of iCalendar objects to be written into the iCalendar file. Each individual iCalendar object is specified by “vcalendar” element type in the file with XML format, and is specified by the strings “BEGIN:VCALENDAR” and “END:VCALENDAR” in the file with iCalendar format. The developed system will work as the standard iCalendar specification and improve the inter-operation with other calendar applications.

6.7 Summary

The preceding sections have recommended and discussed various ways in the project may be developed in the future. Most of new features would be developed in near future. There are of course, several other paths, which the project could follow and there is huge scope for the future development.

Bibliography

1. David Hunter, Kurt Cagle, Chris Dix & Roger Kovack, Beginning XML, 2nd Edition, Wrox Press Ltd pp891-900[16 Dec 2002]

2. http://www.w3schools.com/xml/default.asp [20 Dec 2002]

3. http://training.gbdirect.co.uk/courses/xml/xml_technical_and_commercial_overview_seminar.html [20 Dec 2002]

4. http://omar.opi.arizona.edu/cgi-bin/WebObjects/UAMasterCalendar [20 Dec 2002]

5. http://www.expertsexchange.com/Programming/Programming_Languages/Cplusplus/Q_20419485.html [21 February 2003]

6. Bruce Eckel, 1995, "Thinking in c++", Prentice Hall pp212 [13 Feb 2003]

7. Rick Mercer, 1999, Computing Fundamentals with c++, 2nd Edition, Lynx Communication Group. pp371 [24 February 2003]

8. Bruce Eckel, 1995, "Thinking in c++", Prentice Hall pp221 [08th Mar 2003]

9. http://ietf.org/rfc/rfc2445.txt [15 March 2002]

10. http://www.imc.org/pdi/vcal-10.txt [16 March 2002]

11. http://swordfish.rdfweb.org/calendar/links/ [16 March 2002]

12. http://www.softwarestudio.org/libical/ [18 Dec 2002]

13. http://www.cs.colostate.edu/~cs154/PerfComp [05 Apr 2003]

14. http://www.dfg.cit.cornell.edu/cfg/design/bkg.html [06 Apr 2003]

15. Bruce Eckel, 1995, "Thinking in c++", Prentice Hall p227 [08 Feb 2003]

16. http://gethelp.devx.com/techtips/cpp_pro/10min/2001/aprt/10min0401-2.asp [08th Apr 2003]

17. http://dotnet.di.unipi.it/Content/sscli/docs/doxygen/pal/structtm.html [06 Apr 2003]

18. http://www.ietf.org/internet-drafts/draft-ietf-calsch-many-xcal-02.txt [16 Apr 2003]

Appendix A Project Reflection

Overall I think that the goal of the project was achieved successfully. The original objectives were met and exceeded. However, there are some aspects of the project which could have been better.

I did further research than what I had mentioned in mid-project report. I only talk about what XML format looks like and how XML describe the data in my mid-project report. After researching on relevant documents, I found there is a standard XML is applied to represent iCalendar. Through reading the specification of iCalendar I understood how an iCalendar object is defined and can be identified and parsed within a data stream and what the data format looks line in an iCalendar file. Therefore research can determine the correctness and quality of the project. A good research will direct you to the way of success.

In fact, I did not do as much work at the first semester as I expected, which lead to some changes in the schedule. Fortunately, the original plan was flexible enough to accommodate theses changes, so I had caught up during the holiday. What's more, some unforeseen technical problems were encountered during the implementation of the project. Therefore it is important to devise a project plan that is both well and flexible.

At the early state, the system had ability to import iCalendar file successfully. I considered what methodology I could use to handle XML file. At the beginning, I used an inefficient method to extract information form XML file. Later I learned lexical analyser from the module "Compiler Design". By comparing lexical analyser with the method which was already used, I found lexical analyser was the most powerful, so I decided to use flex to deal with input a stream of XML characters, output a stream of iCalendar characters instead. It took me long time to change code from one methodology to another.

I recorded the advantages and disadvantages of the methods which I used, and also record the differences between the methods if a problem could be solved by many methods. All records reminded me what I had experienced, which was invaluable when I wrote my final project report. Therefore recording the progress of the development is a good habit.

Given the opportunity to conduct a similar project again I would hope to use the experience I have gained to help me to avoid any repetition of these problems, and preserve the good habit to solve some

problem quickly and effectively. I would offer the following suggestions to any students undertaking such a project in the future.

1. Device a flexible plan and Keep to your planned schedule as much as possible. It will give you the opportunity to deal with problems when problems arise. Do not leave anything to the last minute.

2. Always keep a good record of the progress you make during the development. For example, record of what methodology you used, what problems met and how the problems were solved. All of these are invaluable for writing the final project report.

3. Comparing the different methods before implementing carefully and try your best to avoid significant modification of the program. If do so, the time can be saved and the correctness of whole system can be improved.

Appendix B Questionnaire

This questionnaire help me evaluate the software calendar tool I have developed for my project, I would so appreciate it if you would take 15 minutes to use the calendar tool to view and manage your personal information, and give me some your own feelings and opinions on its functions and usability.

If you are a student of school of computing, you can access the system with permission, which is stored in the folder called “project” at my home directory. My username is csxss. Just type ./Interface to run the program, and try each function on the system menu and recode any behaviour that you think is good, bad or unusual, then fill the questions in part A. When you have finished experiencing with all the functionality on the system menu, please fill the questions in part B.

Note: You should specify a date to a scheduling function, which is at least the start date of an event in calendar file, or the scheduling result returned is meaningless. You can schedule file called “calendar.ics”, and specify date “14/04/2003”. Of Course, you can add some your own events and pass a meaningful date to it.

Please e-mail me the completed questionnaire to me at csxss@comp.leeds.ac.uk. In order to meet deadline of the project, I will need your responses by 23rd April.

Part A.

1. View Previous Month Calendar

Was the result correct? yes ☐ no ☐

Was it easy to use? yes ☐ no ☐

Comments: _____ (if any “no”)

2. View Next Month Calendar

Was the result correct? yes ☐ no ☐

Was it easy to use? yes ☐ no ☐

Comments: _____ (if any “no”)

3. View Today's Event(s)

Was the result correct? yes ☐ no ☐

Was it easy to use? yes ☐ no ☐
Comments: _____ (if any “no”)

4. Add New Event

Was the result correct? yes ☐ no ☐

Was it easy to use? yes ☐ no ☐

Comments: too much information need to enter (if any “no”)

5. Edit An Event

Was the result correct? yes ☐ no ☐

Was it easy to use? yes ☐ no ☐

Comments: _____ (if any “no”)

6. Delete An Event

Was the result correct? yes ☐ no ☐

Was it easy to use? yes ☐ no ☐

Comments: _____ (if any “no”)

7. View Full Event List

Was the result correct? yes ☐ no ☐

Was it easy to use? yes ☐ no ☐

Comments: _____ (if any “no”)

8. Schedule Meeting For A Group

Was the result correct? yes ☐ no ☐

Was it easy to use? yes ☐ no ☐

Comments: _____ (if any “no”)

Part B.

1. Were the instructions easy to follow and the system easy to use?

2. Did you ever lost or confused about what was happening?

3. How was your feeling of displaying calendaring information on web browser?

Please give your comments on the whole system and report any errors you have encountered.

The whole system: _____

Sample Questionaire

Part A.

1. View Previous Month Calendar

Was the result correct? yes ■ no ☐

Was it easy to use? yes ■ no ☐

Comments: _____ (if any “no”)

2. View Next Month Calendar

Was the result correct? yes ■ no ☐

Was it easy to use? yes ■ no ☐

Comments: _____ (if any “no”)

3. View Today's Event(s)

Was the result correct? yes ■ no ☐

Was it easy to use? yes ■ no ☐
Comments: _____ (if any “no”)

4. Add New Event

Was the result correct? yes ■ no ☐

Was it easy to use? yes ☐ no ■

Comments: too much information need to enter (if any “no”)

5. Edit An Event

Was the result correct? yes ■ no ☐

Was it easy to use? yes ■ no ☐

Comments: _____ (if any “no”)

6. Delete An Event

Was the result correct? yes ■ no ☐

Was it easy to use? yes ■ no ☐

Comments: _____ (if any “no”)

7. View Full Event List

Was the result correct? yes ■ no ☐

Was it easy to use? yes ■ no ☐

Comments: _____ (if any “no”)

8. Schedule Meeting For A Group

Was the result correct? yes ■ no ☐

Was it easy to use? yes ■ no ☐

Comments: _____ (if any “no”)

Part B

1. Were the instructions easy to follow and the system easy to use?
Yes, generally.

2. Did you ever lost or confused about what was happening?
Yes. When adding an event.

3. How was your feeling of displaying calendaring information on web browser?
Displaying information via the browser is a good idea on the grounds that the information is nice formatted, and is more attractive than pure text based.

4. Please give your comments on the whole system and report any errors you have encountered.
When enter description, if I press enter before type any information, it quits “input description mode” automatically.

The whole system: Major functions are working correctly, program efficiency is generally fine. However, the text based user interface is bit confuse sometimes. For example, when adding an event, there’s a default value pre-set for user who did not input anything for the EVENT START DATE, but there’s no such hints displayed.

Generally, I think the backend program works fine and is tested.

Appendix C Source Code

XML.lex

```
%{
FILE* outfile;
%}

%s CREATEDTIME
%s DATETIMESTART
%s DATETIMEEND
%s TRANS
%s SEQ
%s CLASS
%s SUM
%s LOC
%s CAT
%s DES
%s USERID
%s ALARMSTART
%s ACTION

%%

"<vevent>"                {fprintf(outfile, "BEGIN:VEVENT\n");}
"</vevent>"               {fprintf(outfile, "END:VEVENT\n");}
"<valarm>"                {fprintf(outfile, "BEGIN:VALARM\n");}
"</valarm>"               {fprintf(outfile, "END:VALARM\n");}
"</vcalendar>"            {fprintf(outfile, "END:VCALENDAR\n");}

"<dtstamp>"               BEGIN(CREATEDTIME);
<CREATEDTIME>{
      [^\<]*              fprintf(outfile, "DTSTAMP:%s\n", yytext);
      "</dtstamp>"        BEGIN(INITIAL);
}

"<dtstart>"               BEGIN(DATETIMESTART);
<DATETIMESTART>{
      [^\<]*              fprintf(outfile, "DTSTART:%s\n", yytext);
      "</dtstart>"        BEGIN(INITIAL);
}

"<dtend>"                 BEGIN(DATETIMEEND);
<DATETIMEEND>{
      [^\<]*              fprintf(outfile, "DTEND:%s\n", yytext);
      "</dtend>"          BEGIN(INITIAL);
```

```
}

"<transp>"                  BEGIN(TRANS);
<TRANS>{
      [^\<]*                fprintf(outfile, "TRANSP:%s\n", yytext);
      "</transp>"           BEGIN(INITIAL);
}

"<sequence>"                      BEGIN(SEQ);
<SEQ>{
      [^\<]*                fprintf(outfile, "SEQUENCE:%s\n", yytext);
      "</sequence>"         BEGIN(INITIAL);
}

"<class>"                   BEGIN(CLASS);
<CLASS>{
      [^\<]*                fprintf(outfile, "CLASS:%s\n", yytext);
      "</class>"            BEGIN(INITIAL);
}

"<summary>"                 BEGIN(SUM);
<SUM>{
      [^\<]*                fprintf(outfile, "SUMMARY:%s\n", yytext);
      "</summary>"          BEGIN(INITIAL);
}

"<location>"                      BEGIN(LOC);
<LOC>{
      [^\<]*                fprintf(outfile, "LOCATION:%s\n", yytext);
      "</location>"         BEGIN(INITIAL);
}

"<item>"                    BEGIN(CAT);
<CAT>{
      [^\<]*                fprintf(outfile, "CATEGORIES:%s\n", yytext);
      "</item>"             BEGIN(INITIAL);
}

"<description>"             BEGIN(DES);
<DES>{
      [^\<]*                fprintf(outfile, "DESCRIPTION:%s\n", yytext);
      "</description>"            BEGIN(INITIAL);
}

"<uid>"                      BEGIN(USERID);
<USERID>{
      [^\<]*                fprintf(outfile, "UID:%s\n", yytext);
      "</uid>"              BEGIN(INITIAL);
}

"<trigger>"                 BEGIN(ALARMSTART);
<ALARMSTART>{
      [^\<]*                fprintf(outfile, "TRIGGER:%s\n", yytext);
      "</trigger>"          BEGIN(INITIAL);
}
```

```
"<action>"                  BEGIN(ACTION);
<ACTION>{
      [^\<]*                fprintf(outfile, "ACTION:%s\n", yytext);
      "</action>"           BEGIN(INITIAL);
}

"\n"                  /*ignore the '\n'.*/
%%
```

Appendix D Sample Output Files

Sample iCalendar File

```
BEGIN:VCALENDAR
CALSCALE:GREGORIAN
PRODID:-//Shengqun//Sun Calendar//EN
VERSION:2.0
BEGIN:VEVENT
UID:csxss003-425-811
DTSTAMP:20/04/2003T13:29:38
DTSTART:14/04/2003T00:00:00
DTEND:15/04/2003T00:00:00
TRANSP:OPAQUE
SEQUENCE:4
CLASS:Private
SUMMARY:Sheng's Birthday
LOCATION:Maxi's
CATEGORIES:Personal
DESCRIPTION:6pm at bar of Maxi'
BEGIN:VALARM
TRIGGER:16/04/2003T18:09:57
ACTION:DISPLAY
END:VALARM
END:VEVENT
BEGIN:VEVENT
UID:csxss003-425-811
DTSTAMP:20/04/2003T14:33:52
DTSTART:02/05/2003T15:00:00
DTEND:02/05/2003T16:00:00
TRANSP:OPAQUE
SEQUENCE:3
CLASS:Private
SUMMARY:interview
LOCATION:computer science department
CATEGORIES:personal
DESCRIPTION:see Dr Simon
BEGIN:VALARM
TRIGGER:20/04/2003T14:35:42
ACTION:DISPLAY
END:VALARM
END:VEVENT
BEGIN:VEVENT
UID:csxss003-425-811
DTSTAMP:27/04/2003T20:50:56
DTSTART:05/05/2003T10:00:00
```

```
DTEND:06/05/2003T17:00:00
TRANSP:OPAQUE
SEQUENCE:3
CLASS:Private
SUMMARY:Finish so31 coursework
LOCATION:Long room
CATEGORIES:Education
BEGIN:VALARM
TRIGGER:05/05/2003T08:00:00
ACTION:DISPLAY
DESCRIPTION:so31
END:VALARM
END:VEVENT
END:VCALENDAR
```

Sample XML File

```
<?xml version="1.0" encoding="iso-8859-1"?>
<iCalendar>
<vcalendar version="2.0" prodid="//Shengqun//Sun Calendar">
<vevent>
  <uid>csxss003-425-811</uid>
  <dtstamp>20/04/2003T13:29:38</dtstamp>
  <dtstart>14/04/2003T00:00:00</dtstart>
  <dtend>15/04/2003T00:00:00</dtend>
  <transp>OPAQUE</transp>
  <sequence>4</sequence>
  <class>Private</class>
  <summary>Sheng's Birthday</summary>
  <location>Maxi's</location>
  <categories>
      <item>Personal</item>
  </categories>
  <description>6pm at bar of Maxi'</description>
  <valarm>
    <trigger>16/04/2003T18:09:57</trigger>
    <action>DISPLAY</action>
  </valarm>
</vevent>
<vevent>
  <uid>csxss003-425-811</uid>
  <dtstamp>20/04/2003T14:33:52</dtstamp>
  <dtstart>02/05/2003T15:00:00</dtstart>
  <dtend>02/05/2003T16:00:00</dtend>
  <transp>OPAQUE</transp>
  <sequence>3</sequence>
  <class>Private</class>
  <summary>interview</summary>
  <location>computer science department</location>
  <categories>
      <item>personal</item>
  </categories>
  <description>see Dr Simon</description>
```

```
  <valarm>
    <trigger>20/04/2003T14:35:42</trigger>
    <action>DISPLAY</action>
  </valarm>
</vevent>
<vevent>
  <uid>csxss003-425-811</uid>
  <dtstamp>27/04/2003T20:50:56</dtstamp>
  <dtstart>05/05/2003T10:00:00</dtstart>
  <dtend>06/05/2003T17:00:00</dtend>
  <transp>OPAQUE</transp>
  <sequence>3</sequence>
  <class>Private</class>
  <summary>Finish so31 coursework</summary>
  <location>Long room</location>
  <categories>
     <item>Education</item>
  </categories>
  <valarm>
    <trigger>05/05/2003T08:00:00</trigger>
    <action>DISPLAY</action>
    <description>so31</description>
  </valarm>
</vevent>
</vcalendar>
</iCalendar>
```