ITEC810 Project
An Application Suite for a Student Database

Department of Computing
Macquarie University
Sydney, Australia

Project Supervisor: Abhaya Nayak

Student Id: 41064755
Andrew Johnson

5th June 2009

Abstract

The Macquarie University hosts a portal providing staff members with supportive information for their daily recordkeeping responsibilities and needs. Administrators use this information to determine if a student is on track to complete their course in the allocated time. Access to this information is a difficult and slow process by which time the need for the information may have passed. In this paper I present a solution to provide administrators with online information about their students. The solution is an Application Suite for a Student Database. The data recorded in the student database consists of details such as expected submission (completion) dates; ideal submission dates and current load. The student database will be managed by the administrators. It is the responsibility of the administrators to ensure that all the required information is entered into the system. In the paper I also provide details on the methods and architecture used to deliver this solution. It is also the intention of the paper to show that the final product is easy to maintain and use.

Table of Contents

1. Introduction

Administrators at Macquarie University regularly request information about their students from the Macquarie University Records and Archives Services (RAS) department to assist in making decisions regarding their students. In order to receive this information a formal request needs to be made to the relevant work[1] unit. This request is made either in person, by phone, by email or online by filling in the online file request form. In the request they must state the fields that they require which at this point they may not fully understand as yet. As the person requesting the information may not understand all the fields they can request. They may miss a field containing data that is critical to their needs. They may also request fields that are not relevant and therefore increasing the size of the data unnecessary.

The information is then provided in a data file containing the requested raw data. These files are provided on a monthly basis. There are many problems with this process. Files are very large. To view the information the user needs to load the file through an application such as MS Excel, which can be cumbersome and difficult for some users. The information is not accessible to everyone and because the file contains data on many students as well as containing many different sections of student information some users would need to ask for permission to view the file. This can take up more valuable time.

The user needs to understand all the fields even though they may not be relevant. As this is a very rigid structure there is a lot of redundant data and this makes it much more difficult to notice trends or patterns and therefore makes it impossible to analyse the data and compare the information. Due to the frequency of file delivery the need for the information may have passed before the information can be accessed.

My Application Suite will be accessible on line and therefore provide the user with quick and easy access to enter and view the required student details. The only system requirement is a web browser and access to the Computing Department web-server. Only relevant student details will be stored in the student database thereby making the information provided by the application more meaningful. Customized graphs and statistical reports will allow the administrators to analyse and compare their student's progress in their particular courses to ensure they are on track. A query interface will also allow the user to access specific student details.

[1] Higher Degrees Research Unit (HDRU) for PHD students.

2. Related Work

In researching this project I did not find any existing software package similar to my solution. The current system at Macquarie University is just a request portal and does not provide any facility for staff to manage their student information. It does not provide any real-time access to student details or the ability to produce any summary reports. Even if I did find similar pack-aged software, no matter how customizable, I don't believe it would be able to meet 100% of the needs of this project. Due to inherent limitations of software, we would have to compromise on certain aspects of the software, or even amend the processes of the project to the software. In many cases custom software development is less expensive than a general application because it is designed to meet the business needs [5]. Users will get the software to do exactly what they need it to do, saving time. The application suite has been designed and developed taking into account the much-specialized needs of administrators.

3. System Requirements

In summary the solution needs to provide the ability for administrators to enter student's details into a database, view or export the details and run reports with graphical representation such as bar charts and pie charts. These abilities need to be accessible online through a web browser. The solution also needs to be simple to use and have the capability to handle new student attributes without the need for additional coding from a developer.

Detail requirements listed below:

3.1 The application suite for student data-base (ASSD) shall be available on-line.
3.1.1 ASSD shall be accessible on multiple internet browser platforms.

3.2 ASSD shall provide user name and password verification protection to protect from unauthorized use of the system.

3.3 ASSD shall allow the suite manager to add, remove and modify user names and pass-words are required.

3.4 ASSD shall allow the suite manager to add, remove and modify student attributes as required.

3.5 ASSD shall allow the user to manage the student database.
3.5.1 ASSD shall allow the user to add and delete students.

3.6 ASSD shall allow the user to generate excel reports.

3.7 ASSD shall allow the user to specify bar chart or pie chart preference in the report.
3.7.1 ASSD shall provide the user with a wizard interface to select the required student attribute for report.
3.7.2 ASSD shall allow user to add, modify and delete report templates.
3.7.3 ASSD shall be developed using Active Server Pages interface with Oracle database.

4. The Application Suite

In this section, I provide an overview of the application suite, focusing on the maintainability of the system in a changing environment.

Basically maintainability means we need to have flexibility in design and structure to meet changing needs without redevelopment. One method is to move away from a relational structure to an object orientated structure. This is the method I have adopted to accommodate the changing requirements for the student attributes.

There needs to be a balance between maintainability, dependability and development time allocation. High maintainability increases pressure on dependability as more flexibly leads to increase in the possibility of errors occurring. Both higher maintainability and higher dependability increases development time. I have tried to meet this balance by focusing on the area I feel will have the most amount of change. This being the area around student attributes. The other areas such as report types and security I have left for future work.

Error identification and notification techniques will help with dependability by providing a reliable solution. In this regard I have tried to capture all possible errors through error trapping both in the application modules and the database procedures. This should ensure that the system will not crash when an error is detected but rather handle it internally and notify the user that an error has occurred.

Included in the project scope were hardware and software restrictions. We needed to choose a database platform to use. I have chosen to use Oracle as I have the most experience developing applications with Oracle and it assisted with the limited time I had. There was also the choice of application development software and what web server platform to use. Once again I chose ASP.NET written in visual basic running on Microsoft Internet Information Server as this is where most of my experience lies. Apache web server using Java would have been my alternative choice.

4.1 System Architecture

Portability is one of the key concepts of high-level programming. Portability is the software code-base feature to be able to reuse the existing code instead of creating new code when moving software from an environment to another [3]. The pre-requirement for portability is the generalized abstraction between the application logic and system interfaces [4].

To ensure that there is software portability between different database-platforms I have implemented a modularized solution whereby most of the application suite methods and functions are located in the database. An example of one of these methods and functions is the process to add and remove a system user. All the

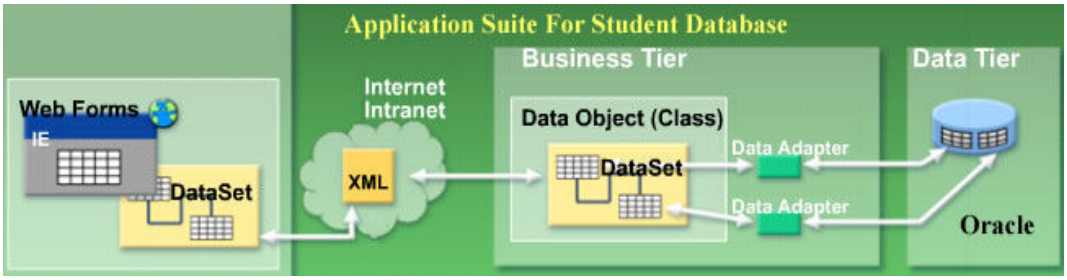

Figure 1: Example of ADO.NET Architecture. This example demonstrates the segregation between the different tiers allowing for portability between the ASP pages, data sets and the database.

code required to perform these tasks is included in the database procedures rather than in the visual basic methods in the ASP page.

The application implements these methods and functions through simple API calls using Active data Objects thereby reducing the amount of code in the application and reducing overhead on the application server. ADO.NET uses XML[2] for all data transport. See Figure 1. This is an advantage because XML has no run-time/transport requirements and therefore no code is required to marshal across the Internet. ADO.NET is a collection of classes, interfaces and structures that manage data access from relational data stores within the .NET framework.

In summary the Advantages of ADO.Net are:

- ADO.NET Does Not Depend On Continuously Live Connections
- Database Interactions Are Performed Using Data Commands
- Data Can Be Cached in Datasets
- Data Is Persisted as XML
- Schemas Define Data Structures

The evolution of Oracle on Windows began in 1993; according to Compuware (2002) today 3.2–6.4 million Visual Studio developers use an Oracle database on their back-end machines [6]. Oracle9i Release 2 is completely .NET-enabled and ensures secure, high-performance, scalable and reliable deployment of enterprise applications in the new Microsoft application and development environment. .NET supports all data access paths such as ODBC.NET, OLE DB.NET or ADO.NET (by the native Oracle Data Provider for .NET).

[2] See http://www.w3schools.com/XML/xml_whatis.asp

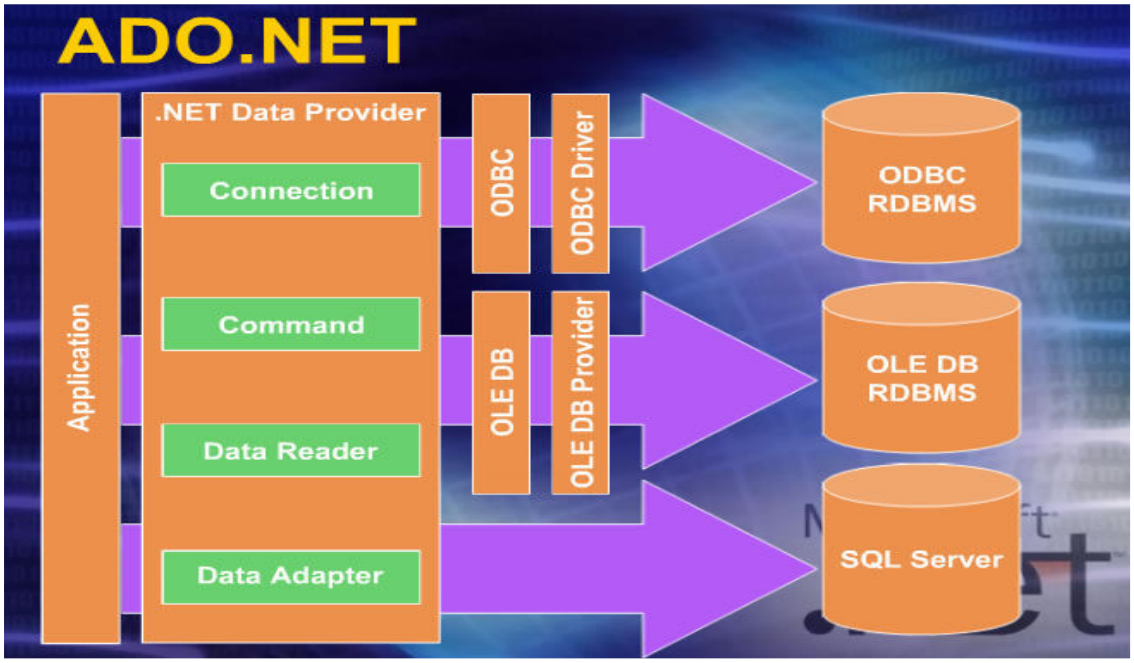

Figure 2: Example of Application portability by switching the Data Providers.

Portability between different databases platforms is realized through switching the Data Providers. The Data Providers that make up the library are specific to a particular database that they would connect to. Oracle Data Provider is used to connect to Oracle or equally SQL Server Data Provider to connect to SQL Server as demonstrated in Figure 2.

4.2 User Interface

The best user interface design guidelines are guidelines in the true sense high level and widely applicable directing principles [1]:

- Know the user population.
- Reduce cognitive load.
- Engineer for errors.
- Maintain consistency and clarity.

The application suite interfaces has been de-signed bearing these guidelines in mind. See Figure 3 and Figure 4 for comparison on consistency and clarity.

Most desktop applications put the commands (such as save, edit, and delete) in fly-out menus at the top of the application. The problem was that on the web, users associate fly-out menus with navigation. They expect commands to appear as buttons or link on the page. In this particular case, users would clearly be frustrated by the system as evidenced by the high volume of "how do I…? calls" received by a client's call centre. My solution here was to use a "button bar" – putting key

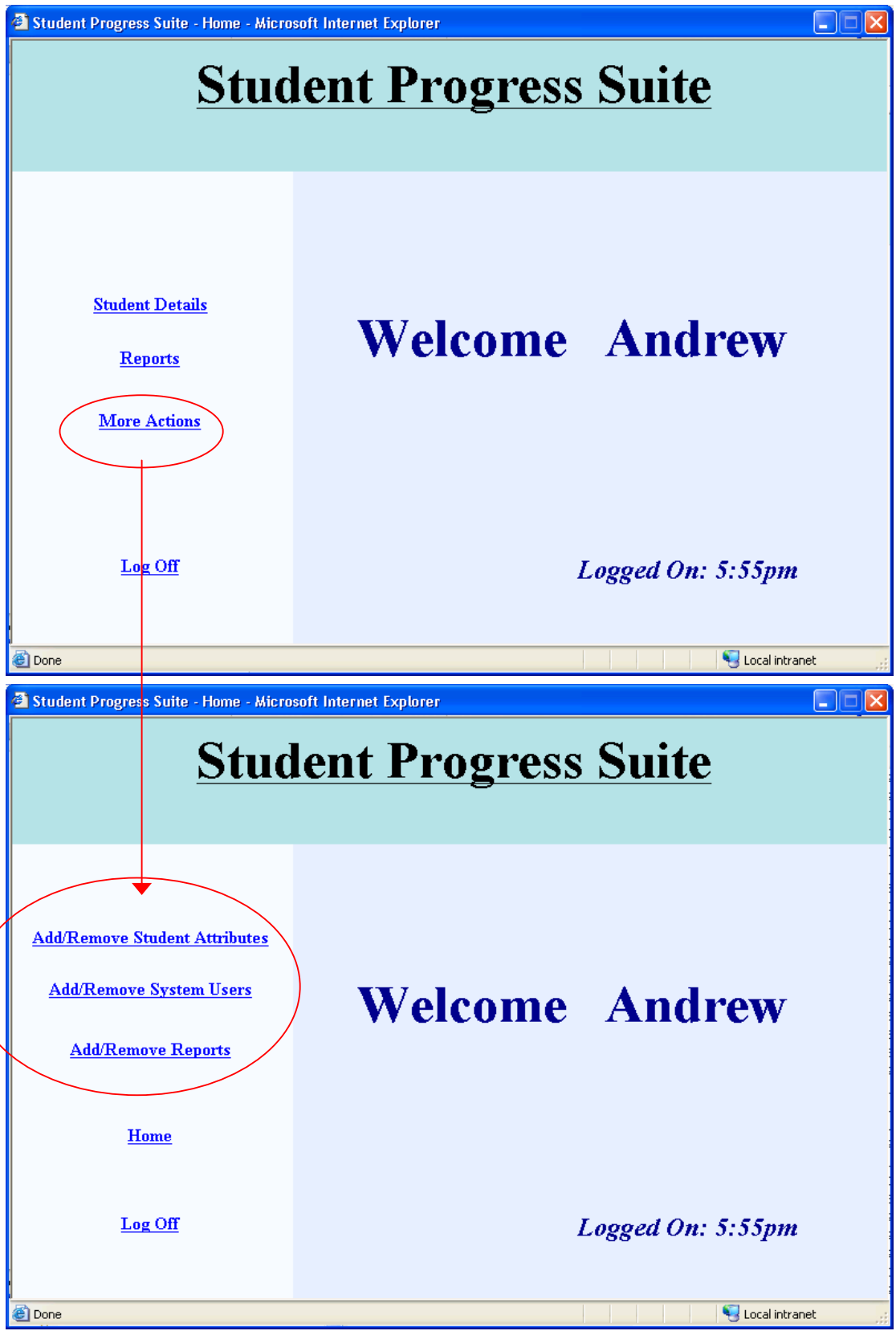

Figure 3: Example of less frequently used commands contained under "More Actions" menu item.

Attribute Master

ATTRIBUTE_ID	**1**
ATYPE	LIST
ANAME	CURRENT LOAD
AVALUES	Part Time; Full Time; Leave; Completed
ADEFAULT	Full Time
ADESCRIPTION	Indicates type of enrolment in course

ATTRIBUTE_ID	**2**
ATYPE	DATE
ANAME	JOIN DATE
AVALUES	
ADEFAULT	Today
ADESCRIPTION	Date student enrolled.

ATTRIBUTE_ID	**3**
ATYPE	NUMERIC
ANAME	NO PUBLICATIONS
AVALUES	
ADEFAULT	0
ADESCRIPTION	Number of publications

Student Master

STUDENT_ID	**1**
STUDENT_NUMBER	41064755
FIRST_NAME	ANDREW
SURNAME	JOHNSON

STUDENT_ID	**2**
STUDENT_NUMBER	41065653
FIRST_NAME	JANE
SURNAME	SMITH

Student Attribute

STUDENT_ID	**1**
ATTRIBUTE_ID	**1**
VALUE	Full Time

STUDENT_ID	**1**
ATTRIBUTE_ID	**2**
VALUE	25-Jan-2008

STUDENT_ID	**1**
ATTRIBUTE_ID	**3**
VALUE	0

STUDENT_ID	**2**
ATTRIBUTE_ID	**1**
VALUE	Completed

STUDENT_ID	**2**
ATTRIBUTE_ID	**2**
VALUE	25-Jan-2004

STUDENT_ID	**2**
ATTRIBUTE_ID	**3**
VALUE	3

Table 1: Student Master and Attribute Master tables with example of two students' details in link table.

commands directly on the page as buttons. Less frequently used commands were contained within a "More Actions" menu as shown in Figure 4. This is an approach that a number of web-based applications have adopted, including Gmail and Yahoo Mail.

When designing the user interface for generating customized reports I could have adopted dragging and dropping as the interaction method. However studies indicate that many users do not expect drag-and-drop to be available on the web [2]. Drag-and-drop can also be quite mouse-intensive, particularly for frequently performed tasks. For this reason, I have implemented a Command based way to move objects from one location to another. So rather than dragging student attributes from the available list to the selected list then user needs to click the add button and remove button to remove the attribute from the selected list.

4.3 Database Structure

One of the project requirements this paper addresses is maintainability. The solution needed to be flexible enough to handle new requirements without further coding. To achieve this requirement I created a table that dynamically stores student attributes rather than creating a table with a fixed set of fields to store each of the students' attributes. This means I treat each attribute as an object with its own properties rather than just a database field. This will allow the system administrator to manage additional student attribute types himself/herself rather than having to call in a developer to add new table fields if new student attribute types are required.

The application suite uses three tables to manage the student attributes; Student Master, Attribute Master and Student Attribute. Student master table contains the unique student id and main student attributes such as the student number, first and last name. Attribute master table contains all relevant attributes definitions that can be associated to a student who has enrolled at Macquarie University. Attributes such as join date, supervisor, ideal submission date, expected submission date etc are all individual entries in the table. Student Attribute table contains the actual attribute value and links it to the student through the student id and links it to the attribute definition through the attribute id.

The Attribute Master table contains all the properties for each attribute such as attribute identifier, attribute type, attribute name, attribute values, attribute default value and attribute description. Attribute type describes the attribute as being Numeric, Boolean or Text. The attribute type can also contain a list of predetermined values. For example attribute current load would contain the list part time, full time, leave of absence and completed. Attributes using the list of predetermined values over a text or numeric type reduces the need for field entry validation as the user is limited to selecting only a valid entry from the list. This will reduce overhead on the system. This is recommended when setting up a new student attribute but may not always be possible. Each attribute as the option for a default value. It is good practice for all numeric attributes to have a default value as the attribute could be used in analytical calculations and a null entry may cause inconstancies or even crash the system. Each attribute has a

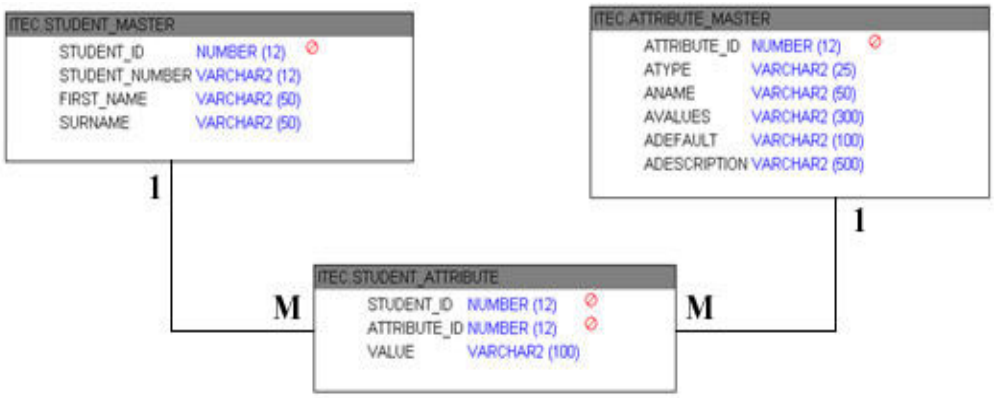

Figure 4: Demonstrates the linking between the three tables managing student attributes.

section to store a brief description about the attribute. This is displayed to the user and will provide a better understanding of the attribute and its use. The relationship between the attributes and students is a many to many relationship. One attribute may belong to many students and one student may have many attributes. The relationships are stored in the student attribute link table as shown in Figure 5. In Table 1 we can see the student 1 (Andrew Johnson) has the following attributes; Current Load, Join Data and Number of Publications. If there is a need to add a new attribute such as Expected Submission Date. The user can simply add a new entry into the attribute master table and then enter the value into the student attribute table for each student respectively. This task will be preformed through the add attribute() method.

4.4 Reporting

In designing Reporting Services, designers are required to meet the following requirements:

- User interactivity is available through parameters, links, conditional display, end-user sorting, drill-down, and a document map.

The user can generate reports through using the report wizard interface. The user is able to simply select the required student attributes they want in the report and then if it is a frequently used report they can save the template which they can use time and time again without having to reselect the attribute each time. These templates may be modified or deleted if no longer required. The reports are generated in MS Excel.

I considered using Crystal reports but there are some disadvantages. The biggest issue with Crystal running on a server is that the crystal components keep leaking memory and it leaves the server crawling after
some time. It is alright with a client-server app where crystal is running on the client side in which case the memory gets released as soon as the client app exits. In the scope of this project installing Crystal Reports on the client is not an option. Apart from that there are issues with deploying Crystal Reports. It works on some systems and fails with some license issue on others.

I have also chosen MS Excel as it provides much more flexibility than other options such as Crystal Reports. Using MS Excel also provides the ability for the user to modify and tweak the report if required.

Student Submission Tracking

Report Date: 05-May-2008

Date Range: 01-Jan-2010 to 01-Jan-2015

Expected Submission		Total	%
Before Time		34	23%
On Time		68	46%
After Time		45	31%
Total Students		**147**	

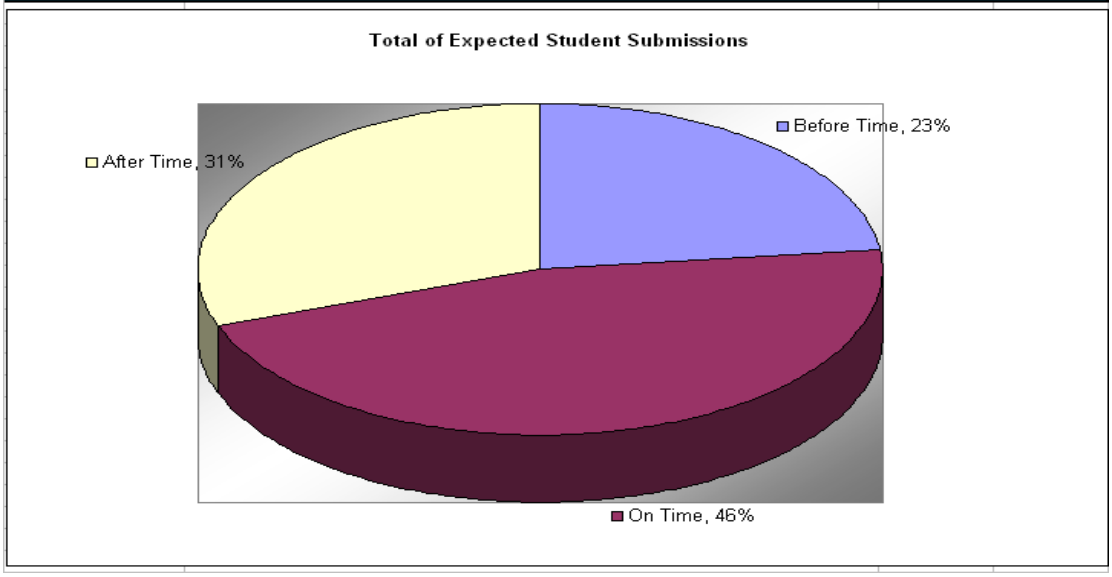

Figure 5: Example of Pie Chart

A chart is a graphical representation of data. In future work the reporting services will offer a wide range of chart formats. Currently the user has the option to select Pie chart (Figure 5) or Bar chart (Figure 6).

I have tried to build a structure that will assist in future work in the reporting module. I have implemented the object-orientated approach here. Object-oriented programming offers a powerful model for writing computer software. Objects are "black boxes" which send and receive messages. This approach speeds the development of new programs, and, if properly used, improves the maintenance, reusability, and modifiability of software. A programmer can simply create a new object that inherits many of its features from existing objects. This makes object-oriented programs easier to modify.

This approach works very well with reporting modules as reports require a lot of repetitive structures and routines. It will allow for quick new report development. Although initially it takes more time to setup it provides a good platform from which to build reports. Through the use of inheritance the developer will save a considerable amount of time developing any new report as most of the code has already been written.

I have created five main classes that are used when coding any of the report formats. So far I have used them when setting up the Pie chart report and Bar chart report. See Appendix A for the details of the five classes.

Student Current Load

Report Date: 05-May-2008

Date Range: 01-Jan-2007 to 01-Jan-2010

Load Type		Total	%
Part Time		25	17%
Full Time		68	46%
Leave Of Absence		10	7%
Completed		45	30%
Total Students		**148**	

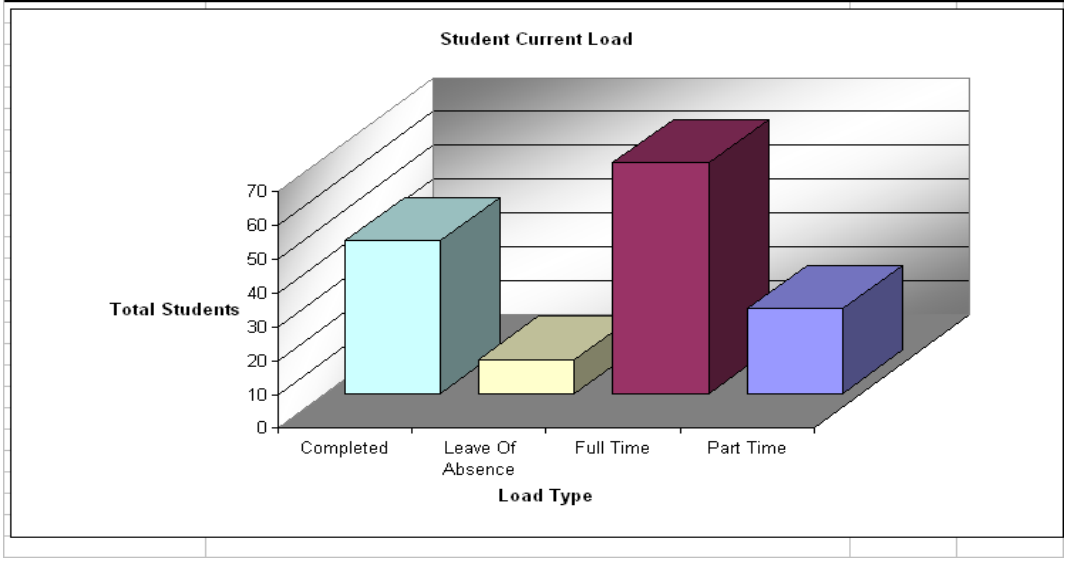

Figure 6: Example of Bar Chart

- The ReportBuilder class sets up the database connection through ADO.NET. It also sets up the error report collection and error logging routine.

- The ExcelReportBuilder class inherits the ReportBuilder. The class creates the excel object when Initialized. The class provides the option to create a new blank excel object or create a new excel object from a template. Designing a creating excel templates can save a lot of development time as it takes more time to write the code to generate a excel sheet from scratch and then format it then if the format is already saved in the template. This does however limit flexibility. Finally the class will save the excel document when finalized and then release all excel objects.

- The Format class contains many predetermined format classes. The classes consist of predetermined values for excel characteristics such as FontSize, FontBold, WrapText, RowHeight, ColWidth and HorizontalAlignment. These characteristics are set when formatting the cells on the worksheet.

- The ExcelPen class is used to enter values into the excel sheet through the Write, WriteFormula and AddFormula methods. These methods implement the desired format from the format class.

- The ExcelTool class is used to find, create and format the excel worksheet. The class has methods to set foreground and background colours. There are methods to control borders and merge cells. I also control the page setup, page footer, zoom and autofit settings using methods from this class. Lastly but still important I have the method to create the charts in this class. Using the method you can specify the chart type, title, data range, row, column, height and width. There are also optional settings such as x and y titles, title fonts and font sizes.

Below I demonstrate how I implement these classes:

```
Public Class ShareholderSurveyReportBuilder
  Inherits ExcelReportBuilder

  Public Sub New(ByVal DBConnection As ADODB.Connection,
                 ByVal ApplicationPath As String)
    MyBase.New(DBConnection, ApplicationPath)
  End Sub
.
.
.
    Private Sub PrintSummaryReportBody(ByVal StatisticsWorkSheet As
                Excel.Worksheet)

    Dim ExcelTool As ExcelTool = New ExcelTool(StatisticsWorkSheet)
    Dim TitlePen As ExcelPen = New ExcelPen(StatisticsWorkSheet,
                                        GetType(ReportTitleFormat))
.
.
.
    TitlePen.Write(Graphics.TitleRow, Graphics.TitleCol, inputReportName)
.
.
.
    Dim ExcelTool As ExcelTool = New ExcelTool(StatisticsWorkSheet)
    Dim ColumnHeaderPen As ExcelPen = New ExcelPen(StatisticsWorkSheet,
                GetType(ShareholderColumnHeaderFormat))

        ExcelTool.ColumnWidth(Graphics.TotalCol) = 1.71
        ColumnHeaderPen.Write(ExcelRowCounter, Graphics.TotalCol, "Total
                              Students")
        ExcelTool.BorderAround(ExcelRowCounter, Graphics.TotalCol,
                              ExcelRowCounter,
                               Graphics.TotalCol + 4,
                              Excel.XlBorderWeight.xlMedium)
.
.
.
```

There are several methods for dynamically filtering report contents:

- Query parameters filter data at the source as it is retrieved.
- Report filters, applied to a dataset or data region, limit the data that is displayed from a generated report.

Using filters retrieves all data, but only data that is relevant to the user is displayed. This may be less efficient on an individual report basis than filtering at the source. However, it lets you retrieve the data once from the source and store in it a snapshot to serve many different user communities. On the other hand, when using query parameters, you must revisit the data source for each new value of the query parameters. In my solution I have chosen to use query parameters. If this was a windows application solution rather than a web based application I may have chosen the filter option. Using the query parameter approach puts all the overhead on the database rather than on the client and reduces the amount of traffic across the network.

The application suite provides a query interface to search and view one or many student details. There is also a feature to export these details in a comma separated text file. This can easily load into excel if required. Future work will include formats such as HTML, PDF, and XML.

4.5 Authentication

Currently there are two levels of user authentication to access the application suite for student database. Firstly the user can only access the site internally and therefore they need to have logged into the Macquarie University network through the normal standard protocols. The second level is a user name and password provided by the application suite system administrator. The password is stored encrypted on the database. A simple encrypt and de-encrypt formula is used by the login method.

To ensure that the user needs to log in before they can access any of application pages I use a signature stored in the active session.

```
Public Const asdbSignature As String = "nGBDB48wefj123bf@3"
Public Const asdbPageDEF As String = "eklrtgnelkth"

        Session.Item("asdbRunning") = True
        Session.Item("asdbSignature") = Global.asdbSignature
        Session.Item("StartBy") = Global.asdbPageDEF
        Pub.NewWebPage(Me, "Home.aspx")
```

When each page is loaded I check in the page load method if the signature has been set correctly and if the page has been started by the correct previous page by verifying the StartBy variable.

In future work we could include one of the follow three authentication methods.

- Windows authentication enables you to identify users without creating a custom page. Credentials are stored in the Web server's local user database or an Active Directory domain. Once identified you can use the user's credentials to gain access to resources that are protected by

Windows authorization. This would be the mostly likely method to implement.

- Forms authentication enables you to identify users with a custom database such as an ASP.NET membership database. Alternatively you can implement your own custom database. Once authenticated you can reference the roles the user is in to restrict access to portions of your Web site.

- Passport authentication relies on a centralized service provided by Microsoft. Passport authentication identifies a user with using his or her e-mail address and a password and a single Passport account can be used with many different Web sites. Passport authentication is primarily used for public Web sites with thousands of users.

4.6 Task Plan

4.6.1 Organization

- Establishment of a regular scheduled meeting with project supervisor to gather data and functional requirements. **Complete.**
- Create and manage a project plan containing detailed tasks, time allocation and milestones. **Complete.**
- Identify risks and draw up a plan to minimise the effects. **Complete.**

4.6.2 Application Design

- A data model depicting the database architecture; Comprising of a data structure diagram and an entity-relationship diagram. **Complete.**
- Prototypes for web-based data entry design. **Incomplete.**

4.6.3 Database Architecture Development

- Develop database structure to store student manage student details. **Complete.**
- Develop database structure to support user access authentication. **Complete.**
- Develop database structure to manage the conflict/error identification, resolution and notification. **Complete.**

4.6.4 Data Management Software

- Develop web-based data entry interface for accepting and storing Student details. **Complete.**
- Develop infrastructure to support Network Authentication. **Incomplete.**
- Develop infrastructure to generate graphical reports. **Complete.**

- User acceptance testing and sign off. **Incomplete.**

4.6.5 End User Access

- Ensure that appropriate security is applied to the data. **Complete.**
- Integrate solution into university domain. **Incomplete.**

4.6.6 End User Support

- Create web-based documentation describing the database architecture and data management processes. **Incomplete.**

4.6 Risks

The success factor of a project can depend on the ability to identify risks. The risk of work commitments can lead to a failure to meet a deliverable milestone. Inaccurate estimates in time allocation is also a risk. The most common being time required to develop the software is underestimated. To increase the success rate of a project we try to assess the probability and seriousness of each risk. For example probability may be low however result effect could be serious.

I ranked the risks in order of probability.

1. Work commitments
2. Underestimating development time
3. Access to university systems
4. Software availability
5. Sickness

During the development there were issues with work commitments. The work commitments limited the amount of time to meet some of the deliverables and resulted in postponement of scheduled meetings with supervisor. The work around was to work later in evenings and take time off from work on non busy work days and reschedule meetings with supervisor.

There was some underestimating of development time but due to experience in the field this has been fairly uneventful.
So far there has been no access to university system. This has not affected most of the project development but could be high risk for final implementation.

The availability of software has not been a problem.

Even though it has been flu season I have been lucky enough not to have caught any sicknesses.

5. Conclusion

In this paper, I described a new application suite aimed at supporting administrators as they facilitate and assist their students through their careers at Macquarie University. The application suite was designed and developed by taking into account the information needs of the administrators. The application suite helps the student administrator determine whether or not the student is on track to complete their course in the allocated time, taking into account total time spent, current load and expected submission date. This is done by providing the student administrator with graphs and statistical reports. This paper also demonstrates the maintainability of the application suite through its dynamic structure and customizable reporting process.

6. References

[1] Jenny Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland and T. Carey. [1994]. *Human-Computer Interaction*. Sydney, New South Wales: Addison-Wesley.

[2] Heidi Adkisson. [2007]. Minimizing Usability Risks in Web Applications. Viewed 3 May 2009.

http://www.blinkinteractive.com/essays/minimizing_usability_risks.php.

[3] Mooney. [1997]. PDF. Bringing Portability to the Software Process. West Virginia University. Dept. of Statistics and Computer Science. Viewed 1 May 2009. http://www.cs.wvu.edu/~jdm/research/portability/reports/TR_97-1.pdf

[4] Garey. [2007]. Software Portability: Weighing Options, Making Choices. The CPA Journal 77 (11): 3

[5] Gabriel J. Adams. [2007]. The Benefits of Custom Software Development vs. Generic Applications. Viewed 3 May 2009. http://ezinearticles.com/?The-Benefits-of-Custom-Software-Development-vs.-Generic-Applications&id=414316

[6] Intel® Business Center Case Study. Viewed 2 June 2009. http://www.oracle.com/technology/tech/windows/odpnet/log-it.pdf

7. Appendix A – Report Module Classes

The following is a sample of the code used to define to report module classes.

7.1 Format Class

```
 Public MustInherit Class FormatToWriteIntoExcel
    Public Sub New()
        NoBackcolor = True
        BackColor = &HFFFFFF
        ForeColor = 0
        Fontsize = 10
        Fontname = "Verdana"
        FontBold = False
        FontItalic = False
        FontUnderlined = False
        VerticalAlignment = Excel.XlVAlign.xlVAlignCenter
        HorizontalAlignment = Excel.XlHAlign.xlHAlignLeft
        BorderLineStyleTop = 0
        BorderLineStyleBottom = 0
        BorderLineStyleLeft = 0
        BorderLineStyleRight = 0
        BorderLineWeightTop = 0
        BorderLineWeightBottom = 0
        BorderLineWeightLeft = 0
        BorderLineWeightRight = 0
        WrapText = False
        RowHeight = 12.75
        ColWidth = 0
        Format = "General"
    End Sub

    Public NoBackcolor As Boolean
    Public BackColor As Integer
    Public ForeColor As Integer
    Public Fontsize As Integer
    Public Fontname As String
    Public FontBold As Boolean
    Public FontItalic As Boolean
    Public FontUnderlined As Boolean
    Public VerticalAlignment As Excel.XlVAlign
    Public HorizontalAlignment As Excel.XlHAlign
    Public BorderLineStyleTop As Excel.XlLineStyle
    Public BorderLineStyleBottom As Excel.XlLineStyle
    Public BorderLineStyleLeft As Excel.XlLineStyle
    Public BorderLineStyleRight As Excel.XlLineStyle
    Public BorderLineWeightTop As Excel.XlBorderWeight
    Public BorderLineWeightBottom As Excel.XlBorderWeight
    Public BorderLineWeightLeft As Excel.XlBorderWeight
    Public BorderLineWeightRight As Excel.XlBorderWeight
    Public WrapText As Boolean
    Public RowHeight As Double
    Public ColWidth As Double
    Public Format As String
End Class
```

```
Public MustInherit Class QueryGeneralFormat
    Inherits FormatToWriteIntoExcel

    Public Sub New()
        MyBase.New()
        Fontname = "Verdana"
    End Sub
End Class

Public Class QueryHeaderFormat
    Inherits QueryGeneralFormat

    Public Sub New()
        MyBase.New()
        Fontsize = 10
        FontBold = True
        RowHeight = 23
        HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter
    End Sub
End Class

Public Class QuerySubHeaderFormat
    Inherits QueryGeneralFormat

    Public Sub New()
        MyBase.New()
        'BackColor = &HFFFF
        NoBackcolor = False
        Fontsize = 10
        FontBold = True
        RowHeight = 23
        HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter
    End Sub
End Class

Public Class QueryColumnHeaderValueFormat
    Inherits QueryGeneralFormat

    Public Sub New()
        MyBase.New()
        Fontsize = 10
        FontBold = True
        WrapText = True
        RowHeight = 23
        ColWidth = 10
        HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter
    End Sub

End Class

Public Class QueryColumnHeaderTextFormat
    Inherits QueryColumnHeaderValueFormat

    Public Sub New()
        MyBase.New()
        HorizontalAlignment = Excel.XlHAlign.xlHAlignLeft
    End Sub

End Class

Public Class QueryBodyValueFormat
```

7.2 ReportBuilder Class

```
Public MustInherit Class ReportBuilder
    Public Sub New(ByVal DBConnection As ADODB.Connection)
        Me.DBConnection = DBConnection
    End Sub

    Public ReadOnly Property ErrorReport() As Collection
        Get
            If fErrorReport Is Nothing Then
                fErrorReport = New Collection
            End If
            Return fErrorReport
        End Get
    End Property

    Protected MustOverride ReadOnly Property TemplateFilename() As String
    Protected MustOverride ReadOnly Property ModuleDescription() As String

    Protected Sub LogError(ByVal Message As String)
        ErrorReport.Add(ModuleDescription & ErrorMessageSeparator &
Message)
    End Sub

    Public ReadOnly Property OutputFilename() As String
        Get
            Return OutputFilenameInternal
        End Get
    End Property

    Public Sub Initialise()
        InitialiseCore()
    End Sub

    Public Sub Finalise()
        FinaliseCore()
    End Sub

    Protected Overridable Sub InitialiseCore()

    End Sub

    Protected Overridable Sub FinaliseCore()

    End Sub

    Protected OutputFilenameInternal As String
    Protected DBConnection As ADODB.Connection

    Private fErrorReport As Collection
    Const ErrorMessageSeparator As String = " - "
End Class
```

7.3 ExcelPen Class

```
Public Class ExcelPen
```

```
    Public Sub New(ByVal Worksheet As Excel.Worksheet, ByVal Format As
Type)
        fTargetSheet = Worksheet
        fFormat = Activator.CreateInstance(Format)
    End Sub

    Public Property TargetSheet() As Excel.Worksheet
        Get
            Return fTargetSheet
        End Get
        Set(ByVal Value As Excel.Worksheet)
            fTargetSheet = Value
        End Set
    End Property

    Public Sub Write(ByVal r As Integer, ByVal c As Integer, ByVal Value As
Object)
        Dim activeCell As Excel.Range
        If fTargetSheet Is Nothing Then
            Throw New Exception("Null TargetSheet - please specify the
ExcelSheet you are going to write on.")
        End If
        activeCell = fTargetSheet.Cells(r, c)
        SetFormatting(activeCell)
        activeCell.FormulaR1C1 = Value
    End Sub

    Public Sub WriteFormula(ByVal row As Integer, ByVal col As Integer,
ByVal Value As Object)
        Dim activeCell As Excel.Range
        If fTargetSheet Is Nothing Then
            Throw New Exception("Null TargetSheet - please specify the
ExcelSheet you are going to write on.")
        End If
        activeCell = fTargetSheet.Cells(row, col)
        SetFormatting(activeCell)
        activeCell.Value = Value
    End Sub

    Public Sub AddFormula(ByVal row As Integer, ByVal col As Integer, ByVal
Formula As Object)
        Dim activeCell As Excel.Range
        If fTargetSheet Is Nothing Then
            Throw New Exception("Null TargetSheet - please specify the
ExcelSheet you are going to write on.")
        End If
        activeCell = fTargetSheet.Cells(row, col)
        SetFormatting(activeCell)
        activeCell.Value = activeCell.Formula & Formula
    End Sub

    Private Sub SetFormatting(ByVal activeCell As Excel.Range)
        activeCell.Font.Name = Format.Fontname
        activeCell.Font.Size = Format.Fontsize
        activeCell.Font.Bold = Format.FontBold
        activeCell.Font.Italic = Format.FontItalic
        activeCell.Font.Underline = Format.FontUnderlined
        activeCell.Font.Color = Format.ForeColor
        activeCell.HorizontalAlignment = Format.HorizontalAlignment
        activeCell.VerticalAlignment = Format.VerticalAlignment
        activeCell.WrapText = Format.WrapText
```

```
        activeCell.RowHeight = Format.RowHeight
        If Format.ColWidth <> 0 Then
            activeCell.ColumnWidth = Format.ColWidth
        End If
        If (Not Format.NoBackcolor) Then
            activeCell.Interior.Color = Format.BackColor
            activeCell.Interior.Pattern = Excel.XlPattern.xlPatternSolid
            activeCell.Interior.PatternColorIndex =
Excel.Constants.xlAutomatic
        Else
            activeCell.Interior.ColorIndex = Excel.Constants.xlNone
        End If
        activeCell.NumberFormat = Format.Format
    End Sub

    Public Sub FuncSum(ByVal TargetRow As Integer, ByVal TargetCol As
Integer, _
        ByVal Row1 As Integer, ByVal Col1 As Integer, _
        ByVal Row2 As Integer, ByVal Col2 As Integer)

        Dim Formula As String
        Dim ExcelTool As ExcelTool = New ExcelTool(TargetSheet)

        Formula = "=Sum(" & ExcelTool.ConvertToR1C1(Row1, Col1) & ":" &
ExcelTool.ConvertToR1C1(Row2, Col2) & ")"
        WriteFormula(TargetRow, TargetCol, Formula)
    End Sub

    Public ReadOnly Property Format() As FormatToWriteIntoExcel
        Get
            Return fFormat
        End Get
    End Property

    Private fFormat As FormatToWriteIntoExcel
    Private fTargetSheet As Excel.Worksheet
End Class
```