An Application Suite for a Student Database

Andrew Johnson
ITEC810
Department of Computing
Macquarie University
Sydney, Australia
andrew.johnson@students.mq.edu.au

Abstract

The Macquarie University hosts a portal providing staff members with supportive information for their daily recordkeeping responsibilities and needs. Student 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 student 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 (part time, full time, leave of absence). The student database will be managed by the student administrators. It is the responsibility of the student 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.

1 Introduction

Student administrators regularly request information about their students from the Macquarie University Records and Archives Services (RAS) department to assist in making decisions regarding their students. To receive this information a formal request is 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. The information is then provided in a data file containing the raw data. These files are provided monthly. 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 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. The user needs to understand all the fields even though they may not be relevant. As this is a much 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 analyze the data and compare the information. Due to the frequency with which the user can receive the file then 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 enter and view the required student details. The only system requirement for the user is a web browser and access to the Computing Department webserver. 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 student administrators to analyze and compare their student progress in their particular courses to ensure they are on track. A query interface will also allow the user to access specific student details.

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

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

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 packaged 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 student administrators.

3 System Requirements

The main requirements for the solution are the ability for the student administrators to enter students and their details into a database, view or export the details of specified students and run reports with graphical representation such as bar charts and pie charts on trends and patterns within the student database. The solution also needed to be simple to use and have the capability to handle new student attributes without the need for additional coding from a developer.

- The application suite for student database (ASSD) shall be available on-line **(3.1):**
    - ASSD shall be accessible on multiple internet browser platforms **(3.1.1)**:
- ASSD shall provide user name and password verification protection to protect from unauthorized use of the system **(3.2)**:
- ASSD shall allow the suite manager to add, remove and modify user names and passwords are required **(3.3)**:
- ASSD shall allow the suite manager to add, remove and modify student attributes as required **(3.4)**:
- ASSD shall allow the user to manage the student database **(3.5)**:
    - ASSD shall allow the user to add and delete students **(3.5.1)**:
- ASSD shall allow the user to generate excel reports **(3.6)**:
- ASSD shall allow the user to specify bar chart or pie chart preference in the report **(3.7)**:
    - ASSD shall provide the user with a wizard interface to select the required student attribute for report **(3.7.1)**:
    - ASSD shall allow user to add, modify and delete report templates **(3.7.2)**:
    - ASSD shall be developed using Active Server Pages interface with Oracle database **(3.7.3)**:

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.

4.1 System Architecture

Portability is one of the key concepts of high-level programming. Portability is the software

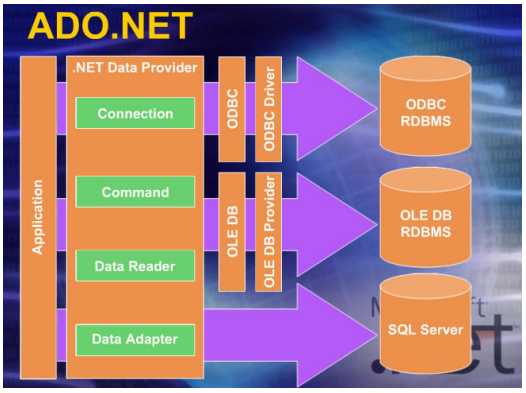


Figure 1: Example of ADO.NET data providers

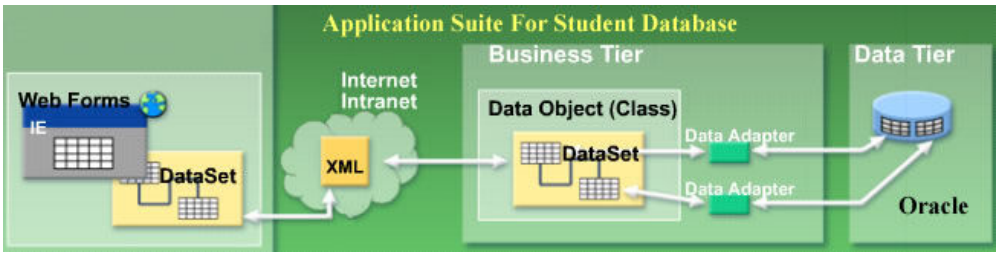


Figure 2: Example of ADO.NET Architecture

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. 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 for all data transport. See Figure 2. This is an advantage because XML has no runtime/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. 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. See Figure 1.

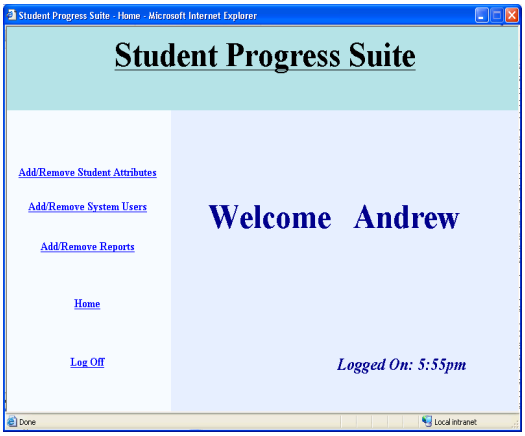


Figure 4: Example of more action items

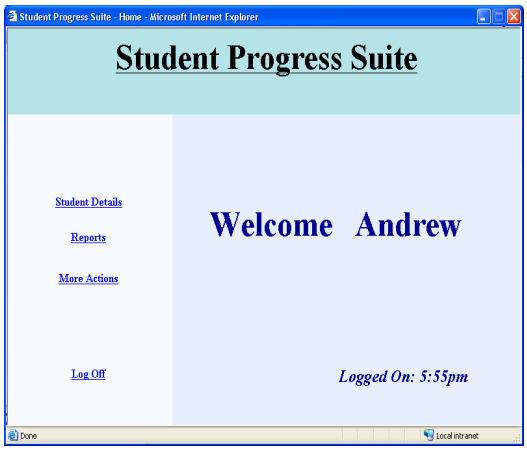


Figure 3: Example of clear and simple interface

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 designed 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 center. My solution here was to use a "button bar" – putting key 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.

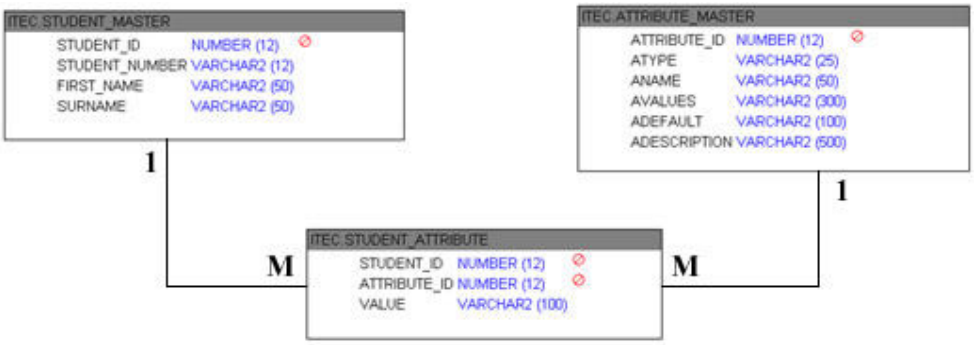


Figure 5: Example of one to many relations between the link & master tables

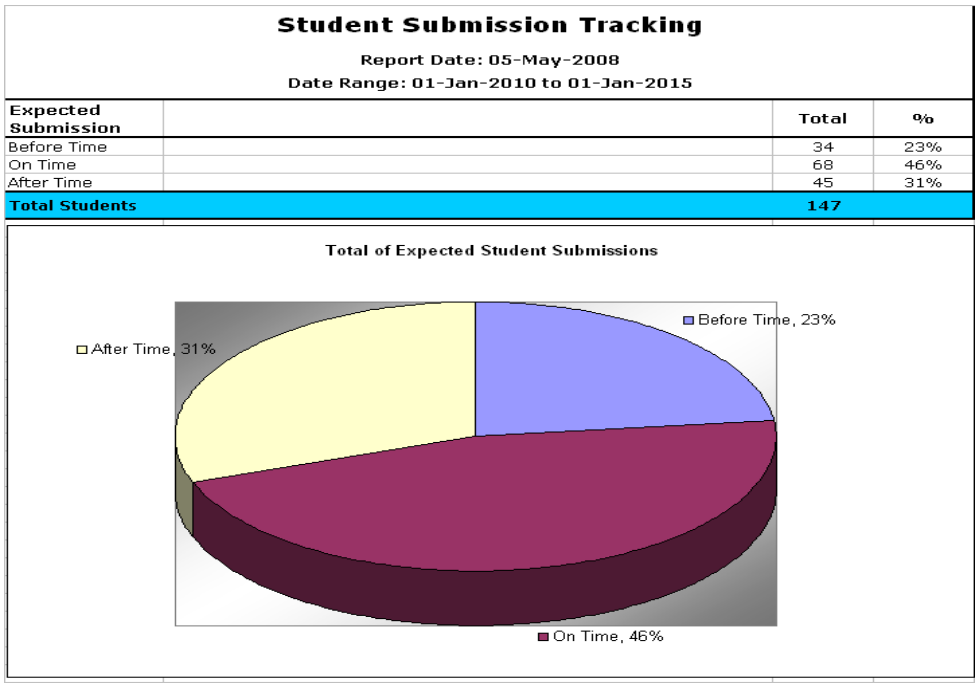
4.3 Database Structure

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 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 table contains the main student attributes such as the student number, first and last name. Attribute master table contains all relevant attributes that can be associated to a student who has enrolled at Macquarie University. Such as program, join date, supervisor, ideal submission date, expected submission date etc. The table has a unique 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 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.

4.4 Reporting

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 in no longer required. The reports are generated in MS Excel. The user has the option to select Pie chart (Figure 6) or Bar chart (Figure 7). I have chosen this

process as MS Excel 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. 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.

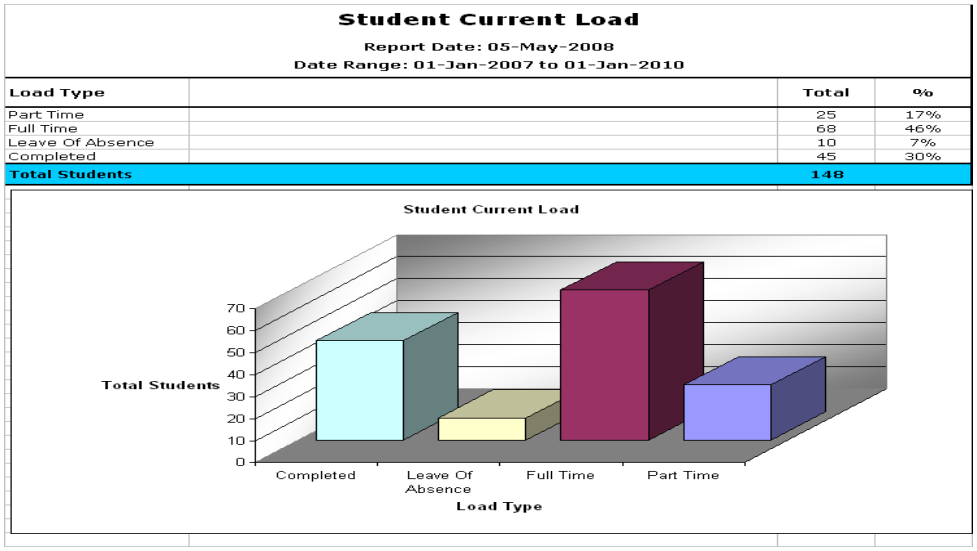


Student Submission Tracking

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 6: Example of Pie Chart of Student Submission Tracking



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 7: Example of Bar Chart of Student Current Load

5 Conclusion

In this paper, I described a new application suite aimed at supporting student 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 student 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.

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.* http://www.blinkinteractive.com/essays/minimizing_usability_risks.php Viewed 3 May 2009.

[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