AN ABSTRACT OF THE REPORT OF

Zhi Wu for the degree of Master of Science in Computer Science of Major presented on March 14th, 2007.

Title: OSU Fee Tracking.

Abstract approved: _____Luca Lucchese_____

Major Professor

This project describes a web-based information management system which has been widely used by staff members and students at OSU. Without this system, it was very hard for the staff members to manage a variety of fees; for example, if they want to change a certain fee, staff members had to send email messages to notify every student and other staff members. For students, it was also hard to find out the latest fees of courses. Now, with the support of this system, it brought a bridge between staff members and students. There are four roles in this project; the first role is administrator who can manage course fees, course fee proposals, departments, and originators and so on. The second role is committee who can browse fees, search fees, and vote confirmation. The third role is originator whose responsibilities are creating course fees, dropping course fees, creating proposal, and withdrawing proposals. The last role is the guest who just can view various fees and download the fee book, but they can not change anything on the web site. With these four integrated roles, we can easily and efficiently manage the fees and streamline the workflow.

This project was realized with the latest Microsoft technologies such as .Net C#, ASP.net, ADO.net, three-tier architecture, SQL Server 2005 and design pattern. It is also managed by the software engineering tools. The milestone and time schedule was set to control the risk of the software development.

OSU Fee Tracking

by
Zhi Wu

A Report

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

March 14th, 2007
Commencement June 2007

Master of Science thesis of Zhi Wu presented on March 14th, 2007

APPROVED:

Major Professor, representing Electrical and Computer Engineering

Director of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

I understand that my report will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Student Name, Author

ACKNOWLEDGEMENTS

I express my thanks to my major professor, Dr. Lucchese for his tremendous support and continuous encouragement. Without his help, this project would not have been possible. I am particularly thankful for his guiding me through this project.

I am grateful to Dr. Nguyen and Dr. Zhang for their willingness to serve on my committee.

I would like to thank the EECS office staff for their patient help in the past two years.

Finally I want to express my special thanks to Mark Clements, who is a senior developer in Microsoft technologies. His guiding led me into this area and helped me become an expert in Microsoft platform and technologies.

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER 1

Introduction

The OSU Fee Tracking system described in this project is widely used by staff members and students at Oregon State University. The purpose of this system is to help staff members manage various fees and to help students view course fees by term. This system has several advantages. First of all, it is very efficient. A person can easily create and change fees. After this is done, other people can see updates immediately through the web. Second, it can prevent errors made by a person. Third, staff members can easily find out the history of fees. Before describing the details of the architecture of this application, I would like to introduce different kinds of fees.

1.1 Fee description

External Fees

External Fees are charged by OSU which provides businesses, organizations, and individuals with goods and services. These organizations or persons do not belong to faculty, staff, students or affiliate organizations. According to University policy and sound fiscal policy, External Fees fully recover the costs associated with providing all goods and services.

Internal Fees

Internal Fees are also charged by OSU which provides members of the OSU community with goods and services. Like External Fees, Internal Fees may keep the 'not to exceed' rules if the unit in setting the pricing policy benefits from the rules. For the Specialized Service Center, additional restrictions may apply. For example, the types of costs determine rates based on actual costs during the course of providing goods and services sold to other departments at OSU. The Internal Fees have the same process as the External Fees for adding, deleting or modifying methods.

Course Fees

Course Fees include charges to students for equipment, materials, or services required as a part of course instruction. The fee applies to two categories: the first one is the equipment or material kept by students as personal property which cannot be bought in the bookstore or through external sources. The second one is the services provided during the course work such as film rental and travel to field sites.

All course fees are supposed to be refundable. The exception is that the case where University is able to recover the cost of the fee incurred by students. For instance, materials that can not be reused or reissued to another student, or a trip discount dependent on the number of participants.

Resource Fees

There are three kinds of Resource Fees: universal fees for all students, program fees for students admitted to particular academic programs and one-time fees for first-term students. The Resource Fees that part-time students need to pay depend on the particular courses taken. Institutions may reset the rates of Resource Fees during the summer session which are different from the preceding academic year's rates. There are three important points to remember about Resource Fees. First of all, a Resource Fee is an Enrollment Fee which applies to students registered in a school, college, department, or degree program. It also applies to the particular classes of students who are all freshmen, sophomores, graduate students and so on or to students in a degree program which falls into business, engineering, forestry, etc. Second, a Resource Fee is not a Course or Laboratory Fee. But for the part-time students, the institution may assess students a resource fee if those students are enrolled in a course provided by a school, college or degree program that assesses a resource fee to an admitted or enrolled student. Third, the number and level of the resource fees is determined by the institutions. According to the procedures for approval of the Academic Year Fee Book, the request for a Resource Fee has to be submitted to a Board of Higher Education for approval.

1.2 Architecture of the system

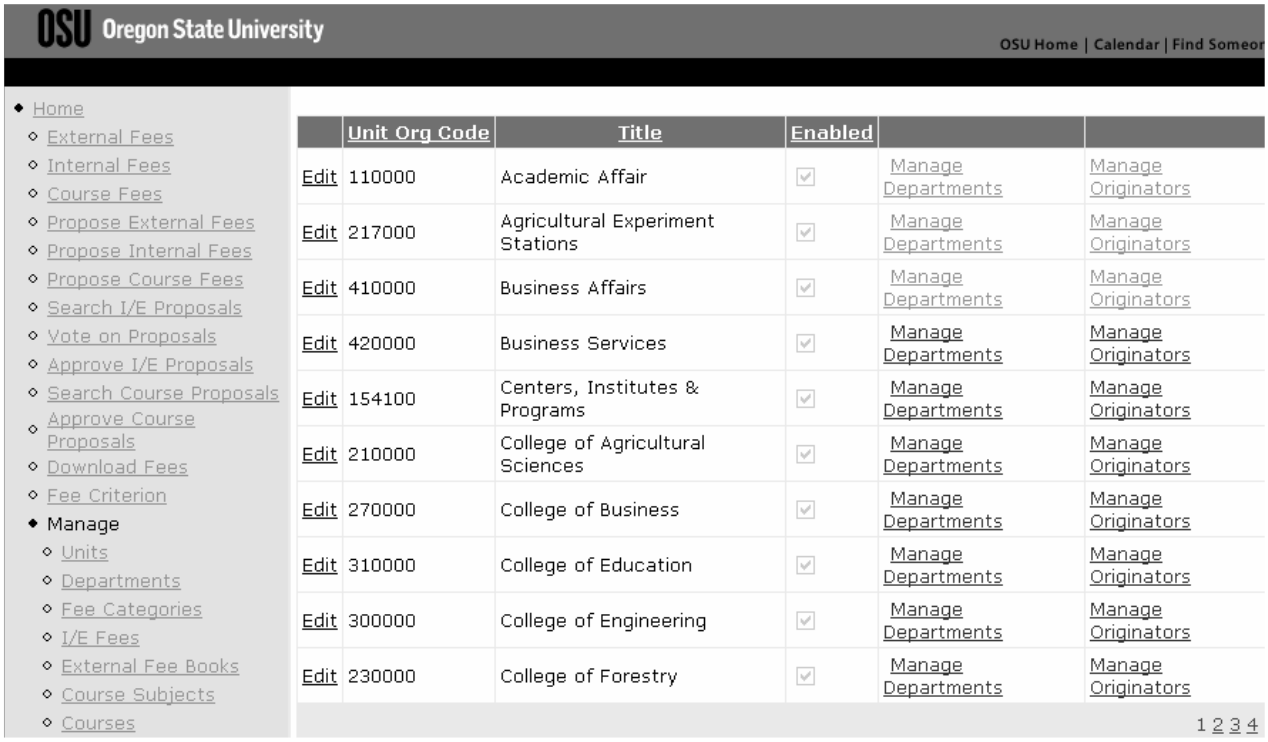

Figure 1: Home page

There are two aspects to the architecture. One is the organization of functionalities of the system, which will be referred to as the soft architecture. The other is the technical architecture of the system, i.e., how the system is realized and what technologies it is based on, which will be referred to as the hard architecture.

1.2.1 Soft Architecture

As can be seen in figure 1, there are two categories of functionalities: ‘Home’ and ‘Manage’. Let’s start from the ‘Manage’ category. In it, the first link is ‘Units’ which represents OSU colleges. When one clicks on the link, the system directs him/her to the

‘Units’ page; in this page, all OSU colleges are listed. At the bottom of this page, one can add a new college if necessary. The ‘Department’ link is similar to the ‘Unit’ link; the difference is that one has to select a college in the ‘Unit’ drop down list on the top of this page, and then the grid view lists all the departments under the unit. The ‘Fee Categories’ link is meant to create a category for a specific fee. Before one can insert a category, he/she has to select a unit and a department and fill in the title and the sequence number in order to insert a category. With the ‘I/E Fees’ link, one can create an internal or an external fee. The ‘I/E Fees’ has the same ‘Unit’ and ‘Department’ as the ‘Fee Categories’ link. He/she has to select a unit and a department first, and check the Internal or External box. After these fields are filled in, he/she needs to click the ‘insert’ link to insert it. For the ‘External Fee Books’ link, it lists the entire fee book versions. In the grid view, he/she can see the pencil and printer icons, which can edit and print the current fee book. Under the ‘External Fee Books’ link, it is the ‘Course Subjects’ link. The courses are grouped by course subject. If one wants to add a new course fee, he/she has to click on the ‘Course Fee’ link, select a subject and a course, fill out the title, amount and detail code areas, and click on the insert link. Clicking on the ‘Terms’ link directs him/her to the term grid view which lists all the terms. Not everybody has the originator’s permission. An originator has the privilege to create a ‘course fees’ or an ‘I/E Fees’. Through the originator’s security link, one can add new originators to the system.

Going back to the ‘Home’ category, one can see the first three links which are ‘External Fees’, ‘Internal Fees’ and ‘Course Fees’. They have a similar layout. Before one can view them, he/she has to select a ‘Unit’ and a ‘Department’ first, and then the page lists particular fees. Clicking the view icon, the system directs one to the detail page of a fee. In the same page, he/she can see the history of that fee if it has a history. Before the ‘External Fee’, ‘Internal Fee’ and “Course Fee’ can be created, they have to be proposed by an originator first. The three proposal links will guide him/her to create proposals. After a proposal is created, committee members can vote for the proposal using the ‘Vote on Proposals’ link. Administors can see the results of voting and decide if he/she should approve the proposal or not. One can download the PDF file of fees through the ‘Download Fees’ link. A guest can see the fee criterion through the last link. The above paragraph described the main functions of this system we created.

1.2.1 Hard Architecture

- Source Control

The OSU Fee Tracking project is based on the team foundation server. The source code is hosted on the server. No matter which terminal he/she is using, he/she can use the 'Source Safe' as the source control tool to retrieve the entire project from the server. With the support of the source control tool, they can work on the same project simultaneously. If two developers are working on the same part, when they check in, they may get a conflict of the source code. The source control tool can help him/her to merge the code together. The team foundation server provides supervisors with other advantages to manage the resource code. When developers are working on the same project, supervisors may need to assign different tasks to different developers. Through the team foundation server, everybody can see active tasks and the progress of current tasks. It helps him/her communicate efficiently.

- Database

SQL Server 2005 is used in the project. The convenient GUI of the database helps him/her easily create the schemas, tables, stored procedures and functions.

- C# Language

The built-in language C# on .Net platform is a very powerful language to develop not only web applications but also Windows desktop applications, networking software, and testing and so on. C# offers several advantages. First of all, pointers have been removed, and he/she does not need to care about the memory management. Second, it possesses some new features such as property, which can be used to get the information in a class securely. Third, one can use C# language and .Net platform to create the web service in only a fewl steps. The advantages are too many to be enumerated one by one.

- Three-tier architecture

The bottom layer of the system is the database layer. In this layer, a database, stored procedures and functions are included.

The middle layer is the 'Business Object Layer'. Many business object classes are written to communicate between the 'Data Access Layer' and the 'View Layer'. Once the 'View

Layer' requests the data, the 'Business Object Layer' helps retrieve the data from the database. Once the view layer saves the data, the 'Business Logic Layer' also helps push the data back to the database.

The top layer is the view layer. Users view, change, and store the data through it. This layer is composed of many controls such as checked box, drop down list, form view, grid view, and so on.

1.3 Organization of the report

The next chapter provides the requirements for the project and introduces the roles and responsibilities of users in the system. The third chapter presents the architecture of the system and the tools used in the implementation of the project. Chapter 4 discusses the implementation details of this project. Chapter 5 presents the testing methods. The final chapter concludes the report by outlining some possibilities for future enhancements.

CHAPTER 2

Project Requirements

The OSU Fee Tracking project involves four roles which are administrator, committee, originator and guest. Each role has its own level of permission and responsibilities. Briefly, the administrator has the largest permission to control the whole process such as adding and deleting other users. The main duty of committee members is to check and vote for proposals. The originator is responsible for creating new proposals. The public guest can only view and download some information from the web site.

2.1 User Roles and Responsibilities

The four roles in the system are not independent. They have their own permissions and responsibilities. At the same time, they have to cooperate with each other to create a certain fee. Why does this system use roles to manage the processes? The answer is simple; the purpose is to simplify complex procedures to let users only manage their own work.

2.1.1 Administrator

The administrator plays the main role in controlling the process. One of the administrator's responsibilities is to check the proposals and approve or deny the proposals. Once the proposals have been created by an originator, the committee members are supposed to vote for each proposal. Administrators can check the result and make the decision to approve or deny a proposal according to a certain policy. After proposals have been approved, administrators go to create the fees such as 'I/E Fees' and 'Course Fees'. Other responsibilities include creating and deleting some basic information such as 'Units', 'Departments, and 'Fee Categories'. An administrator also has permission to add or delete other roles in the system. For example, if a department hires a new employee who joins the originator group, the administrator can add the new person and grant a specific permission to him/her. Vice versa, when an employee resigns himself/herself, there is no reason to keep this role in the system for security reasons. The

administrator will delete the person from the system in time. Administrators would inspect the system every day to guarantee that the system is running correctly.

2.1.2 Originator

The originator is responsible for creating proposals. The information of fees changes from time to time and new fees need to be added. In these cases, an originator should create a new proposal to reflect the changing fee. An originator is supposed to go through the following as described. Once he/she clicks on the proposal link, the originator's contact information is filled in the contact page. If there are no errors, he/she goes to the next page which is the "Fee General' page. In it, he/she can define some general information for current fees. After this page is done, he/she goes to the fee detail page in which he/she specifies the fee type and fee amount information. In the last phase of the process, a summary of the fee is listed. If everything is correct, he/she can save the proposal in the system.

2.1.3 Committee

After the originator has created the proposals, fees cannot be created by administrators immediately. Many committee members have to vote for a specific proposal in terms of the policy at Oregon State University. If most committee members have approved the proposal, it can pass when the administrator checks it. If most committee members deny the proposal, it cannot be approved by the administrator.

2.1.4 Guest

Guests are the users who do not need to log into the system. They can only view and download some information from the web site such as 'Fee Books' and fee criterion information. Guests have limited permission. They can not see the information about proposals and I/E fees. Moreover, they also cannot create and save any information in the system.

2.2 Objective

The purpose of the project described in this report is to manage the fee information in an efficient and convenient way. With the aid of such system, staff members can easily manage fee information and students can see fee changes immediately. When one wants to search the history of a fee, he/she can just click on several buttons to finish these kinds of tasks without searching through piles of documents. Since there are many formats, dropdown lists, checked boxes and forms which are used to limit the information entered, the system can prevent a lot of errors from happening. The other advantage of this system is that it offers a platform to communicate between staff members and students. We are also thinking about adding more functions to the system in the future. For example, we can add a forum to this web site where students can leave messages about their concerns. When the staff members see the messages, they will be able to answer the questions immediately.

CHAPTER 3

System Architecture

In this chapter, we present the architecture of the system. First we present our design pattern for the system and its pros and cons. Then, we outline the detailed architecture and then provide an overview of the Data Access Layer and Business Object Layer.

3.1 Pattern

- A design pattern describes a proven solution to a recurring design problem, placing particular emphasis on the context and forces surrounding the problem [9].

3.1.1 Overview

Model View Controller is a popular Object Oriented pattern and is presented in Figure 2. It is a design pattern that enforces the separation between the input, processing, and output of an application. To this end, an application is divided into three core components: the model, the view, and the controller. Each of these components handles a discreet set of tasks.

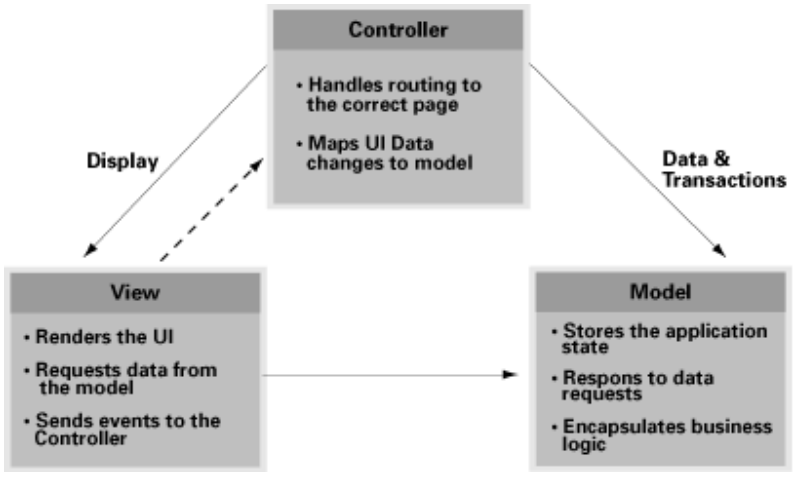

Figure 2: MVC

The view is the interface the user sees and interacts with. For Web applications, this has historically been an HTML interface. HTML is still the dominant interface for Web apps,

but new view options are rapidly appearing. These include Macromedia Flash and alternate markup languages like XHTML, XML/XSL, WML, and web service.

Handling all of these interfaces in your application is becoming increasingly challenging. A big advantage of MVC is that it handles the use of many different views for your application. There's no real processing happening in the view; it serves only as a way to output data and allow the user to act on that data, whether it is an online store or an employee list.

The next component of MVC, the model, represents enterprise data and business rules. It's where most of the processing takes place when using MVC. Databases fall under the model, as do component objects like EJBs and ColdFusion Components. The data returned by the model is display-neutral, meaning that the model applies no formatting. This way, a single model can provide data for any number of display interfaces. This reduces code duplication, because model code is only written once and is then reused by all of the views.

Finally, the controller interprets requests from the user and calls portions of the model and view as necessary to fulfill the request. So when the user clicks a Web link or submits an HTML form, the controller itself doesn't output anything or perform any real processing. It takes the request and determines which model components to invoke and which formatting to apply to the resulting data.

So, to summarize, a user request is interpreted by the controller, which determines what portions of the model and view to call. The model handles interaction with data and applies business rules and then returns data. Finally, the appropriate view is determined and formatting is applied to the resulting data for presentation.

3.1.2 Advantages

Most Web applications are created with procedural languages such as ASP, PHP, or CFML. The initial temptation is to mix data-layer code, such as database queries, in with your display code, which is usually HTML. Separation of data from presentation is something that most experienced developers strive for, but often, this emerges out of trial

and error and not a deliberate plan. MVC enforces the separation of these portions of an application. While it requires extra effort up front to architect an MVC application, the payoffs are impressive.

First, and possibly most important, is the ability to have multiple views that rely upon a single model. As I mentioned, there's an increasing demand on new ways to access your application. A solution to this is to use MVC. With MVC, it doesn't matter if the user wants a Flash interface or a WAP one; the same model can handle either. Code duplication is limited because you've separated the data and business logic from the display.

Because the model returns data without applying any formatting, the same components can be used and called for use with any interface. For example, most data might be formatted with HTML, but it could also be formatted with Macromedia Flash or WAP. The model also isolates and handles state management and data persistence. For example, a Flash site or a wireless application can both reply on the same session-based shopping cart and e-commerce processes.

Because the model is self-contained and separate from the controller and the view, it's much less painful to change your data layer or business rules. If you switch databases, say from MySQL to Oracle, or change a data source from an RDBMS to LDAP, you need only alter your model. If written correctly, the view won't care at all whether that list of users came from a database or an LDAP server. The three parts of an MVC application are black boxes whose inner workings are hidden from the other portions. It makes you build well-defined interfaces and self-contained components.

The concept of the controller is also a benefit. To me, the controller is used to stitch together different pieces of the model and the view to fulfill a request. This places significant power into the architect's hands. Presented with a number of reusable building blocks in the model and the view, the controller picks and chooses which blocks are needed to handle specific processing and display requirements.

3.1.3 Disadvantages of MVC

Among the drawbacks of using MVC is that it's not necessarily easy, and it is definitely not for everybody. MVC requires significant planning. And it introduces a deeper level of complexity that requires diligent attention to detail.

You'll have to spend a good amount of time thinking about the way the parts of your application will interact. Also, the rigorous separation between the model and view can sometimes make debugging more difficult. Each piece will require thorough testing before it can be introduced. The silver lining here is that once a part is tested, you get to reuse it with abandon.

In my experience, using MVC also means having more files to manage than you would otherwise. This might seem obvious, since the whole point is to separate the three MVC areas, but the additional work of dealing with multiple files shouldn't be underestimated. Just keep it in mind.

The bottom line is that MVC is overkill for small applications and even many medium-size ones. The extra work is probably not worth it: Taking the time to lay down a complex architecture may not be worth the payoff.

3.2 Detailed Architecture

To incorporate MVC pattern in .Net, a set of tools and frameworks are used. Particularly, I want to introduce the 'Data Access Layer' and 'Business Logic Layer' in the system. Figure 3 shows the diagram for the architecture.

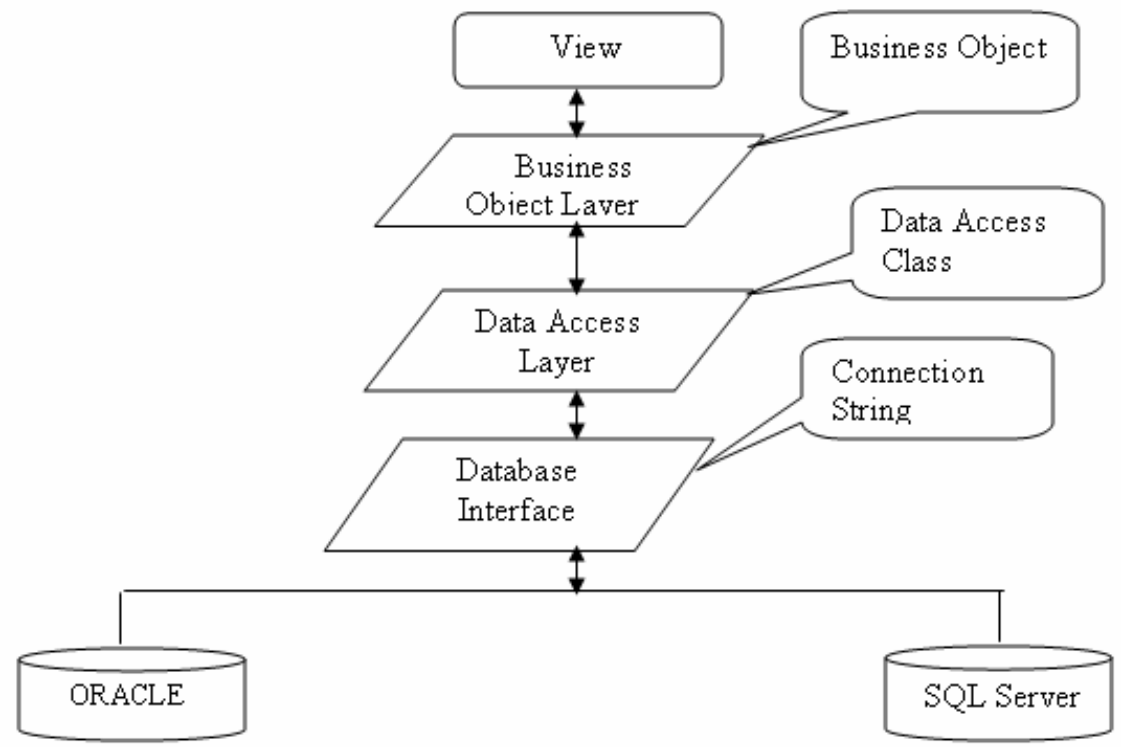

Figure 3: Architecture diagram

3.3 Layers

3.3.1 Data Access Layer

3.3.1.1 Introduction

The ‘Data access layer’ exists between the ‘Business Object layer’ and the database layer. It plays a primary role in retrieving or pushing back data from or to database. Why do we need to use the layer to retrieve and push back data? This method has been approved as a good way to manipulate data. There are many advantages. First of all, it provides flexibility. Usually, the layer includes methods such as inserting, deleting, updating and selecting. Through these functions, he/she can decide which strategy they should use and which specific object they should create. Second, if he/she encapsulates all functionalities into a black box, when errors happen, it is very difficult to find out where the problems are. With the support of the layer, he/she can write testing code to test methods to find out bugs. Figure 4 shows the Selecting method in the ‘Course DA’ data access layer.

```
public static Course Select(string subjectCode, string courseNumber)
{
    SqlConnection conn = OSUFeesDB.GetConnection(false);
    SqlCommand cmd = new SqlCommand(CourseCommand.Select, conn);
    cmd.CommandType = CommandType.StoredProcedure;

    cmd.Parameters.AddWithValue("@CourseNumber", courseNumber);
    cmd.Parameters.AddWithValue("@SubjectCode", subjectCode);

    Course course = null;

    try
    {
        conn.Open();

        SqlDataReader dr = cmd.ExecuteReader(CommandBehavior.SingleRow
        if (dr.HasRows)
        {
            dr.Read();
            course = FillItem(dr);
        }
        dr.Close();
```

Figure 4: Data Access Layer

From the code, two parameters, 'SubjectCode' and 'CourseNumber', are passed to the 'Selecting' method. Then the method establishes the connection to the database through 'SqlConnection' object. 'SqlCommand' calls the stored procedure in the database and binds the two parameters with those in the stored procedure. Later, the 'SqlCommand' retrives data from the database and populates data to the business object. In the layer, he/she can control the process and data by our code. If any problems happen, he/she can test the code in the layer through the values of variables or writing testing code to test the method. From this perspective, bugs will be found out much more easily than using built-in framework in which he/she can not check or test the code.

3.3.2 Business Object Layer

3.3.1.1 Introduction

The Business object layer exists between the 'GUI' and the 'Data Access Layer'. The data access layer is responsible for retrieving or pushing back data from or to the database, but it cannot control how data are displayed in the 'GUI'. The 'Business Object

Layer' is functional as the business logic role in the architecture. Basically, its primary responsibility is to control the access of data. The OOP concept is adopted to encapsulate the data access. Not every method has permission to access the data in business objects. The business object encapsulates data in many 'Private' fields. It also creates 'Public' properties as the interface to access the data. If some outside methods need to access data, they cannot access them directly. The only way to access data is to call interfaces. In this way, he/she can guarantee that data is consistent and secure.

```
private string _subjectCode;
private string _courseNumber;
private string _title;
private bool _enabled;
private string _creditInd;
private decimal _creditHigh;
private decimal _creditLow;
private CourseFeeCollection _fees = new CourseFeeCollection();
#endregion

#region Properties


public string SubjectCode
{
    get { return _subjectCode; }
    set { _subjectCode = value; }
}

public string CourseNumber
{
    get { return _courseNumber; }
    set { _courseNumber = value; }
}
```

Figure5: Business Object Layer 1

For example, from the Figure 5, one can see that the '_subjectCode' is defined as the private field. This means that the other methods can not access 'SubjectCode' directly. In the 'Properties' area, the 'SubjectCode' property is defined as a public interface, in which the 'get' method is responsible for retrieving data from the 'Data Access Layer', and the 'set' is responsible for pushing data back to the database. If a certain method needs to

access the data in the business object, calling this public property is the only way to access it.

The example above only demonstrates how to control the access to data, but does not illustrate how to control the logic.

```
public bool CanBeChanged
{
    get
    {
        //The course fee version cannot be edited if there is an active proposal or if we are w
        //registration start date for the effective term. To check that subtract 14 days from t
        //see if it is greater than the current date.
        //If the effective term is null return false
        if (this.EffectiveTerm == null)
            return false;
        //otherwise check the registration start date to see if our timing is correct
        else
        {
            return ((!HasActiveProposal) && (this.EffectiveTerm.DeadlineDate >= DateTime.Now));
        }
    }
}
```

Figure 6: Business Object Layer 2

Figure 6 presents an idea for controlling the permission to change a particular fee. From the code, the logic checks if the 'EffectiveTerm' is null or not. If it is null, he/she does not have permission to change the fee. If it is not null, the logic checks if the proposal is active and the 'DeadLineDate' is later than 'Now'. If both of the two clauses are true, that means he/she has the persmission to edit the course fee version. This example is only one of thousands of examples of using business object layers to control the business logic. The Business Logic Layer gives he/she more freedom to integrate the logic into the application.

CHAPTER 4

Implementation

In this chapter, we discuss the details of the implementation so that one can have a big picture of how this system is designed. It is not necessary to go through all of those processes since some of the processes are similar. Several processes are picked out to illustrate some functionalities of the system.

4.1 Administrator

4.1.1 Unit management

Administrators can edit and add 'Units' through the 'Units' link. When an administrator clicks on the 'Unit' link, the system directs him/her to the page as in Figure 7. In the page, if an administrator finds that there is something wrong in the record, he/she can click the 'Edit' link in the first column to correct a particular field.

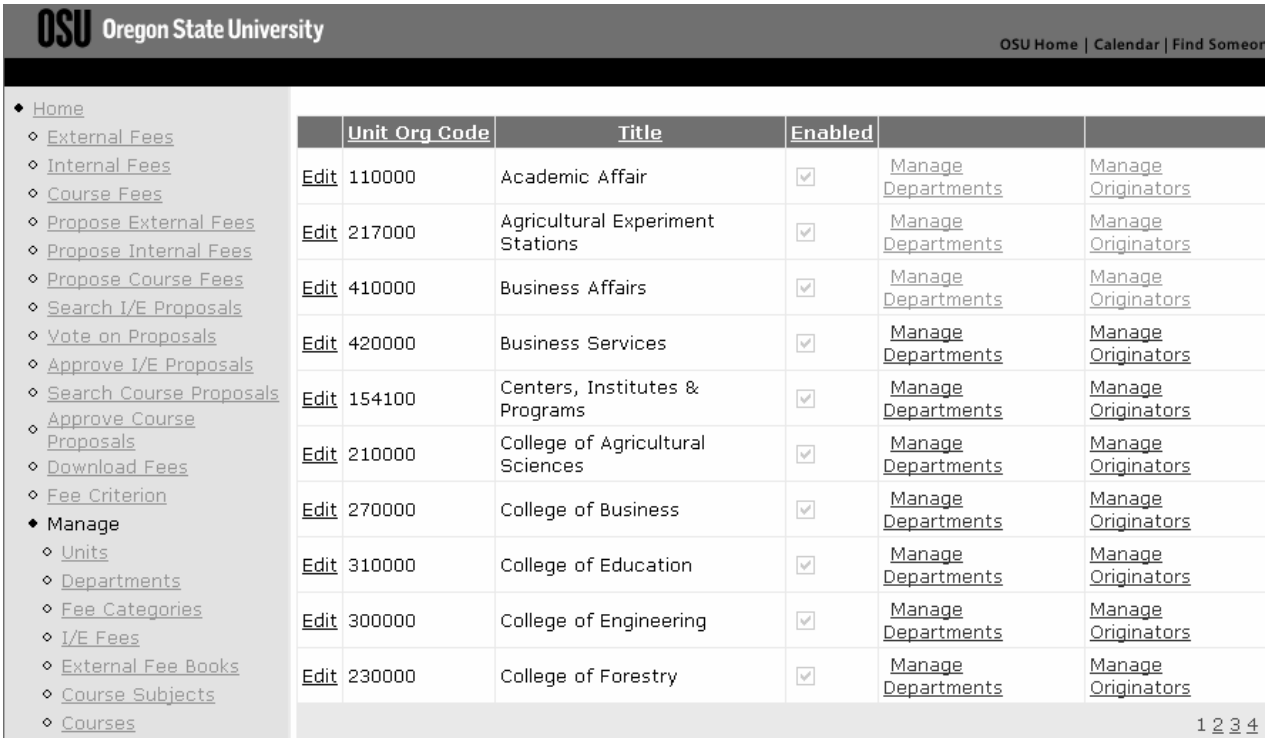

Figure 7: Units page 1

After clicking, he/she is directed to the page as in Figure 8.

	Unit Org Code	Title	Enabled		
Update Cancel	110000	Academic Affair	☑	Manage Departments	Manage Originators
Edit	217000	Agricultural Experiment Stations	☑	Manage Departments	Manage Originators
Edit	410000	Business Affairs	☑	Manage Departments	Manage Originators
Edit	420000	Business Services	☑	Manage Departments	Manage Originators
Edit	154100	Centers, Institutes & Programs	☑	Manage Departments	Manage Originators

Figure 8: Units page 2

One can see that some fields of the record are highlighted by rectangles. The administrator can change the content in rectangles and click on ‘Update’ to save the change; on the other hand, he can click on “Cancel’ to undo what he/she did.

The administrator can add a ‘Unit’. First, the system directs him/her to the page in Figure 9.

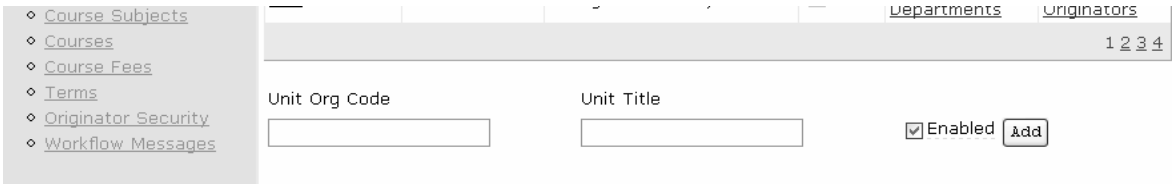

Figure 9: Units page 3

Suppose he/she wants to add a new ‘Unit’ called ‘engineering’; he/she must fill out the two text boxes which are ‘Unit Org Code’ and ‘Unit Tile’ and click on the add button. After these operations, he/she should see that one ‘Unit’ has been added to the grid view as in Figure 10.

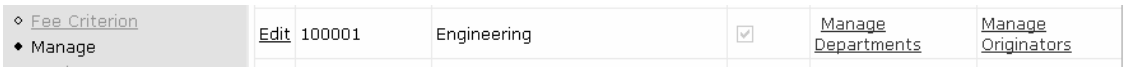

Figure 10: Units page 4

4.1.2 Course fee management

An administrator can add a new course fee by clicking on the ‘Course Fees’ link on the left side of the home page. The system directs him/her to the page as in Figure 11.

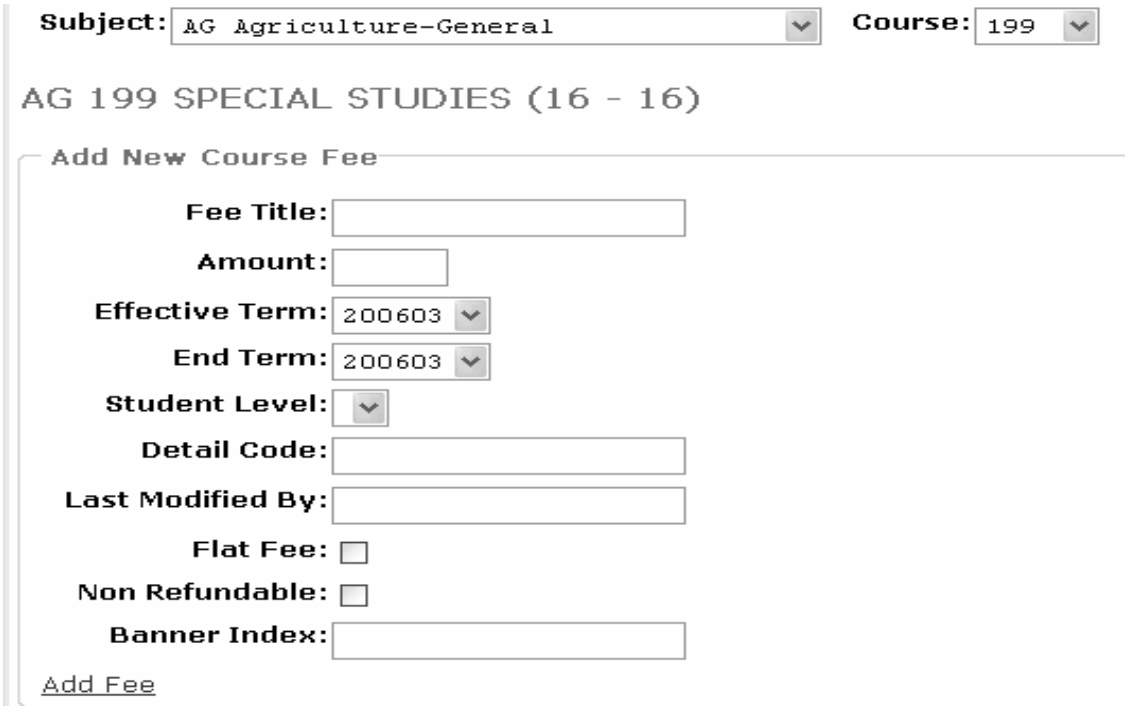

Figure 11: Course Fee 1

In the subject drop down list, he/she chose ‘AG agriculture’ as the subject and ‘199’ as the course number. Then he/she filled in ‘Computer’ for the Fee Title, ‘100’ for the Amount and all of other required fields. Clicking on the ‘Add Fee’ link directs him/her to the page as in Figure 9.

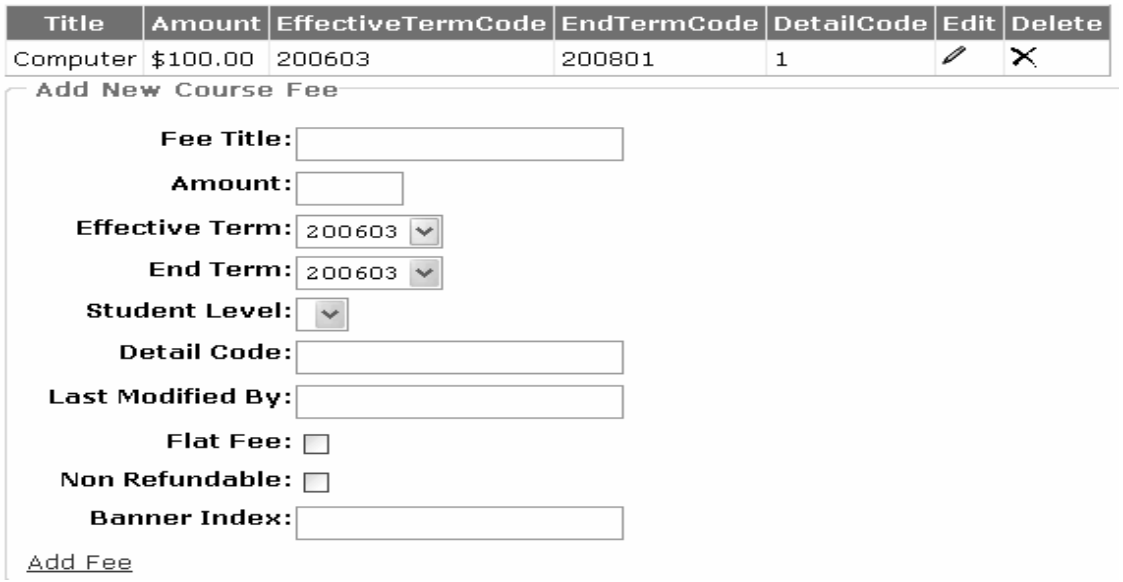

Figure 12: Course Fee 2

On the top of the page, one can see that the fee has been added; the administrator can delete or edit that fee through clicking the delete or edit icon in the last two columns.

4.1.3 Search I/E proposals

An administrator can search proposals through the conditions he/she defined. If he/she wants to view all proposals, he/she has to click on the search button. All proposals are listed in the page as in Figure 13.

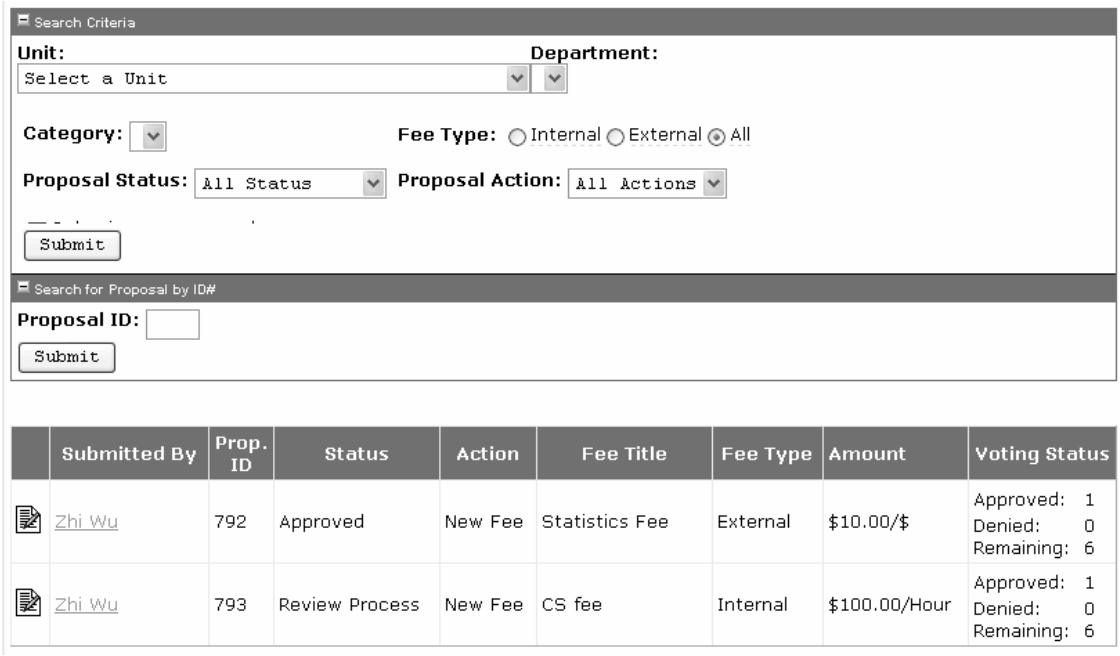

Figure 13: Proposals

4.1.4. Approve proposals

An administrator can approve a particular proposal. First of all, he/she has to define the conditions for the proposal. Second, after he/she gets a proposal, he/she can check the radio button called ‘Approve’ and click on the ‘Submit’ button. The proposal is approved as in Figure 14. After these actions, if he/she wants to check whether the internal fee has been added, he/she has to click on the ‘Internal Fees’ link on the left of the home page. At the bottom of the ‘Internal Fees’ page, he/she can see the ‘CS Fee’ we just added as in Figure 15.

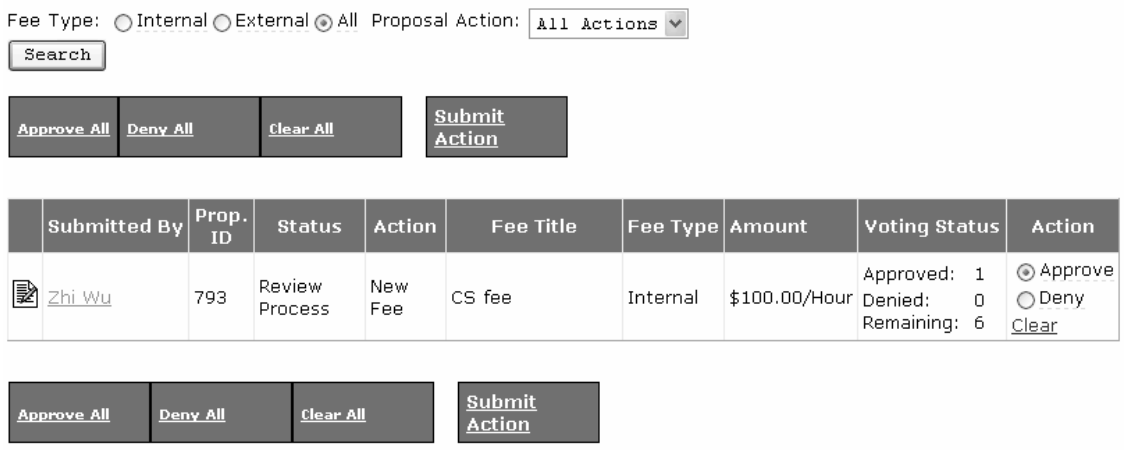

Figure 14: Approve Proposal

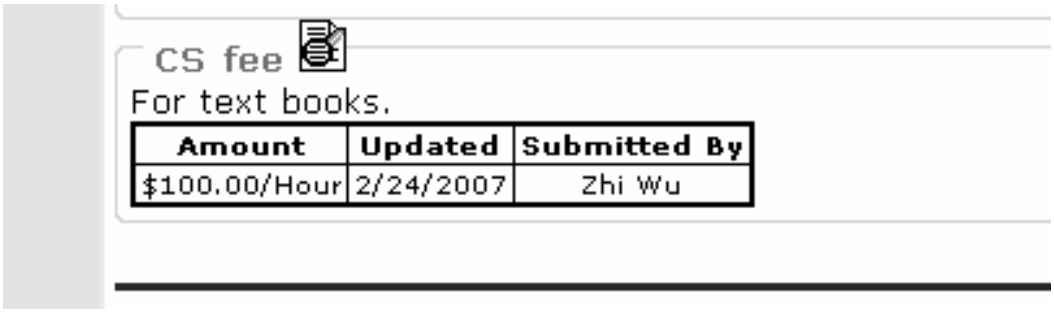

Figure 15: Internal Fee

4.2 Committee

4.2.1 Vote for proposals

After proposals have been submitted, committee members can vote for a specific proposal.

The voting page is same as in figure 16.

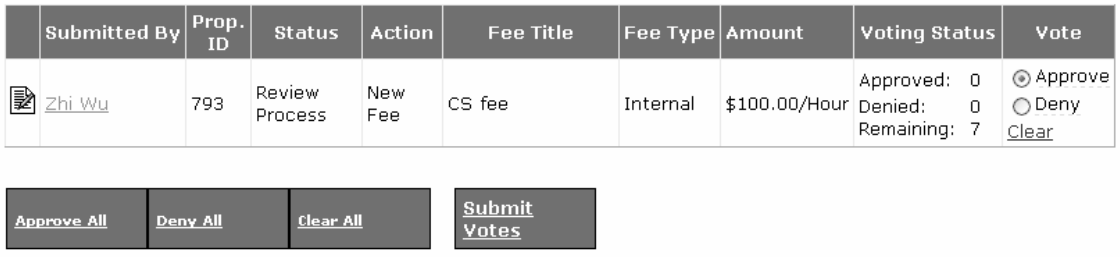

Figure 16: Vote Proposal

Choosing ‘Approve’ and clicking ‘Submit votes’ mean that the committee has approved the proposal. He/she also can choose ‘Deny’ to deny a proposal.

4.3 Originator

4.3.1 Propose I/E fee

An originator can propose an ‘Internal Fee’ only through a wizard. First, he/she clicks on the ‘Propose Internal Fees’ link on the left of the home page. Then he/she chooses the ‘College of Engineering’ as the ‘Unit’ and the EECS as the ‘Department’. Clicking on the ‘New Fee’ in this category directs him/her to the page as in figure 17.

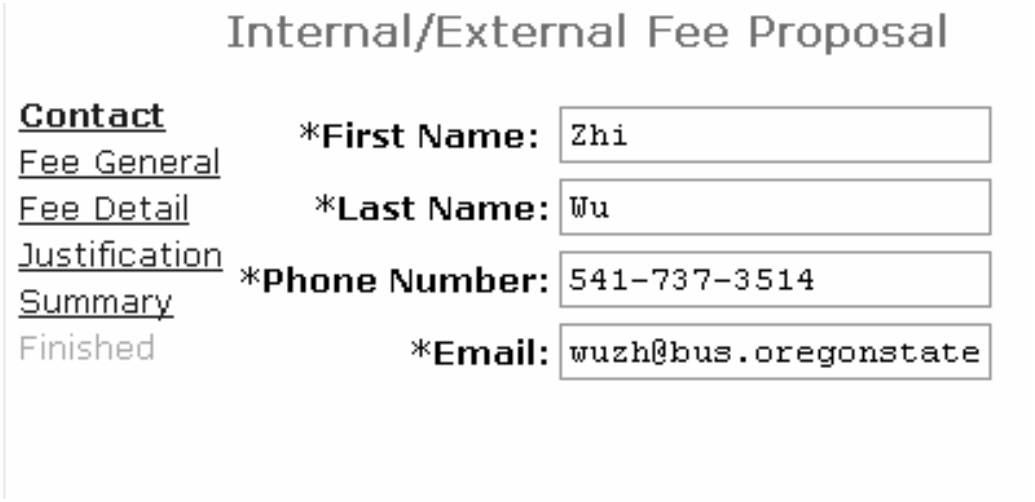

Figure 17: I/E fee 1

He/she has to fill the information in the text boxes with * at the front. Clicking the ‘Next’ button directs him/her to the next page as in figure 18.

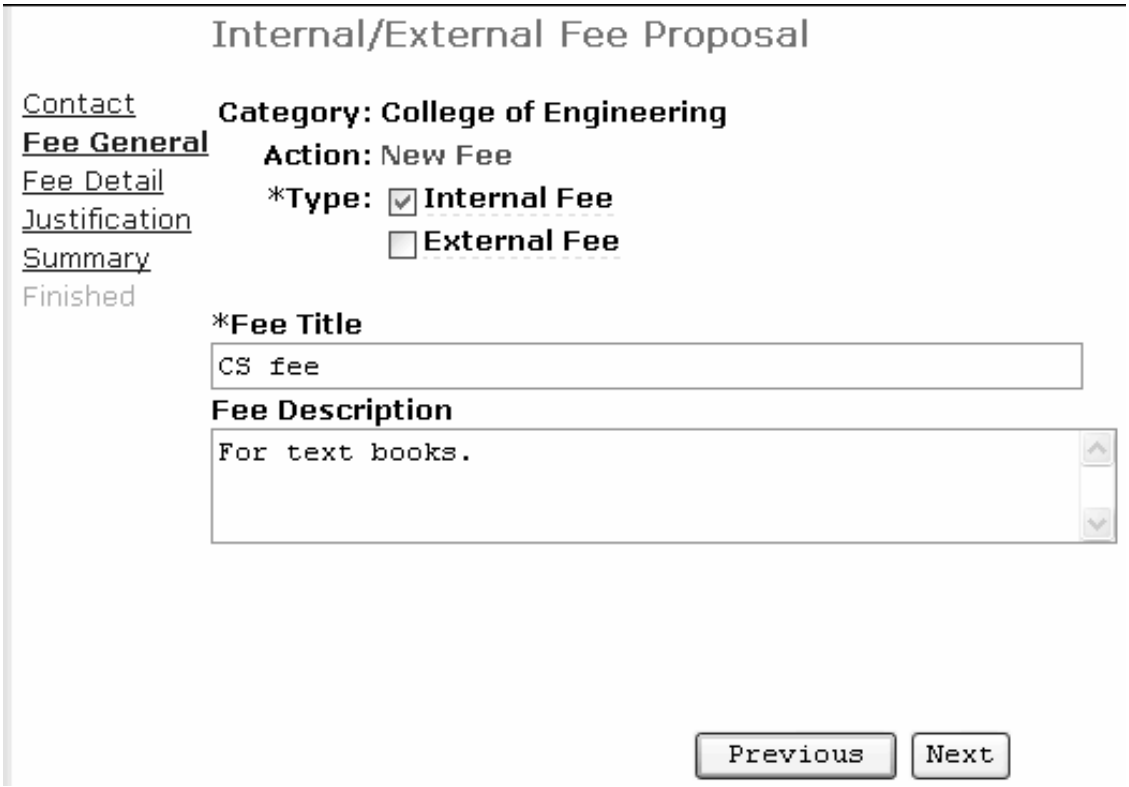

Figure 18: I/E fee 2

In this page, he/she checks the 'Internal Fee' and fills in the 'Fee Title' and 'Fee Description'. Clicking on the 'Next' button directs him/her to the page as in figure 16.

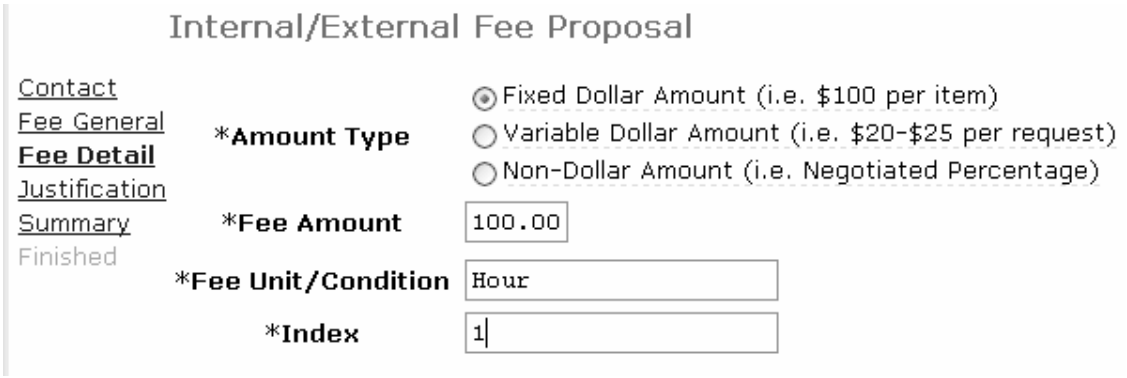

Figure 19: I/E fee 3

He/she checks the first radio button and fills the information in the 'Fee Amount', 'Fee Unit' and 'Index' areas. The system directs him/her to the page as in figure 20 by clicking on the 'Next' button.

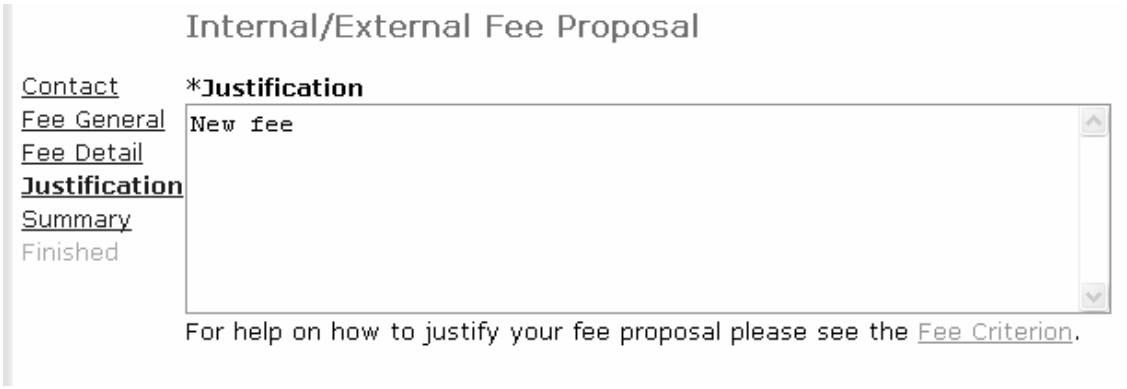

Figure 20: I/E fee 4

After these steps, he/she gets the summary page as in figure 18 which lists all the information he/she just filled in.

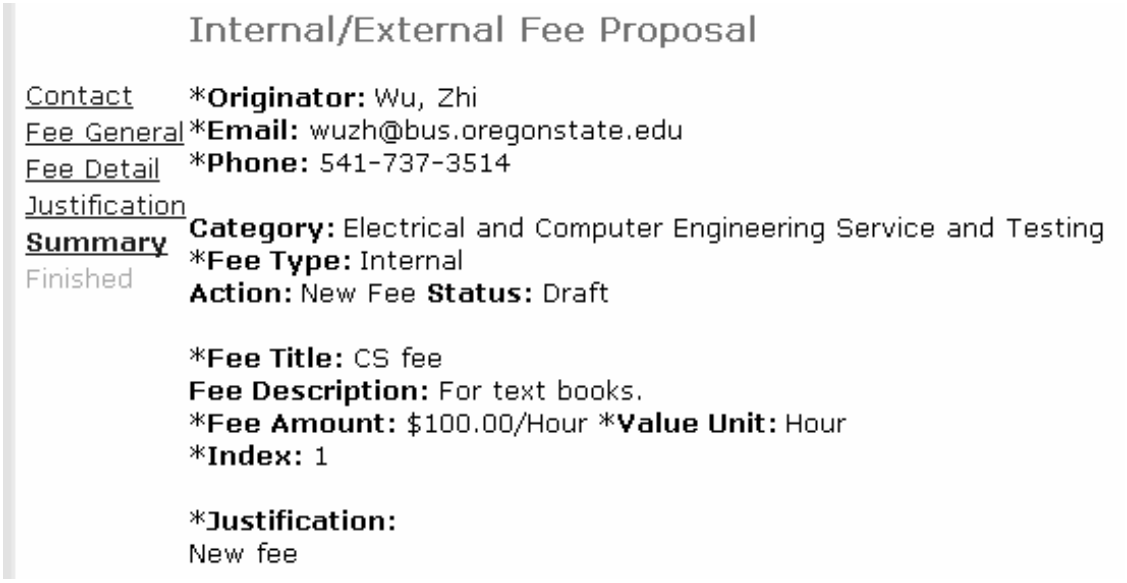

Figure 21: I/E fee 5

After he/she clicks on the 'Submit for review' button, the proposal is submitted and saved.

4.4 Guest

A guest can only view the information on the web site. For example, he/she can view the 'Course fees' and 'Fee Criterion'. If he/she clicks on the 'Course Fee' link on the left side of the home page, the system directs him/her to the page as in Figure 22.

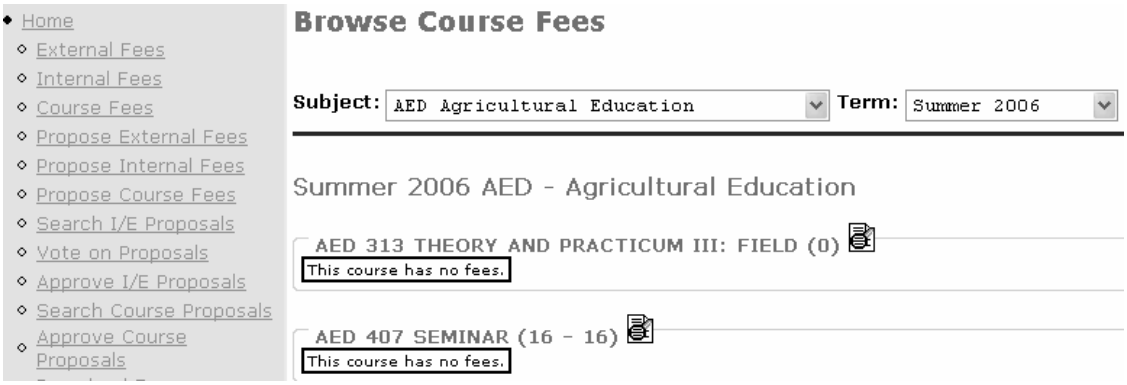

Figure 22: Guest page 1

He/she can see that under the 'AED' subject and 'Summer 2006' term, many classes are listed. If he/she clicks on the 'Course Fee Detail' icon on the 'AED 509' course, it directs him/her to the page as in Figure 23.

Figure 23: Guest page 2

In this page, one can see the detail of the course.

A guest can download the 'Fee Books' as a PDF file. If he/she clicks on the 'Download Fees' link, it directs him/her to the page as in Figure 24.

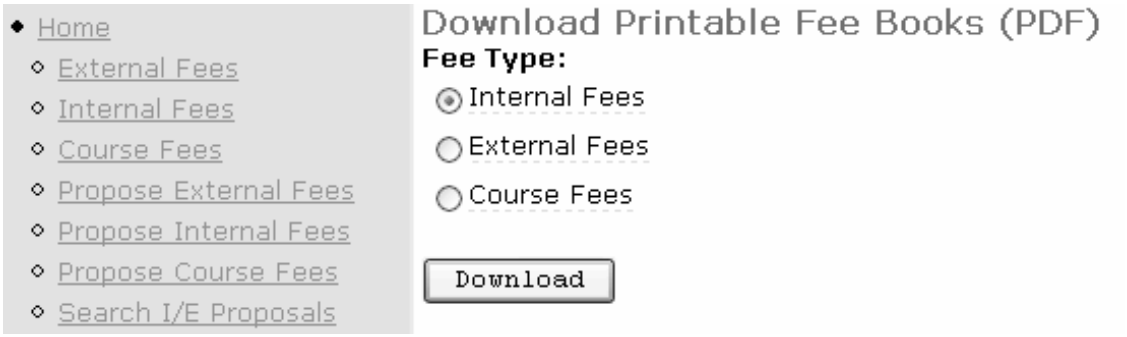

Figure 24: Guest page 3

He/she can choose which fee he/she wants to download. For example, if he/she chooses 'Internal Fees' and clicks on the 'Download' button, he/she downloads a PDF file including various fees.

CHAPTER 5

Testing

Testing is a very important phase in the software development life cycle. The best software is created not only by the best developers but also by the best testers. In .Net platform, the most popular testing technologies are manual tests and unit tests with respect to black box and white box tests.

5.1 ‘NUnit’ Introduction

‘NUnit’ [8] is a testing framework to test functionalities. It allows us to write test code in the same language as our code written for the application. For example, one can write the test code using C#, VB.net, J#, etc. It also provides a testing environment which automates the process of running the tests and checking on the results. ‘NUnit’ takes advantage of the idea from the version of ‘JUnit’ which is the framework for Java. However, ‘NUnit’ has been completely integrated into the .net framework in order to increase its power testing power.

Before writing testing code, one must import ‘Nunit Framework.dll’ into the central ‘NUnit’ library. Then one can use a series of attributes to mark the testing code. One can take advantage of these attributes to organize the code into groups called suites. When running the testing code, the test runner executes according to these attributes.

5.2 Implementing the manual test (Black Box Test)

‘Manage Course’ web page in Figure 25 is a good example that illustrates what the manual test is.

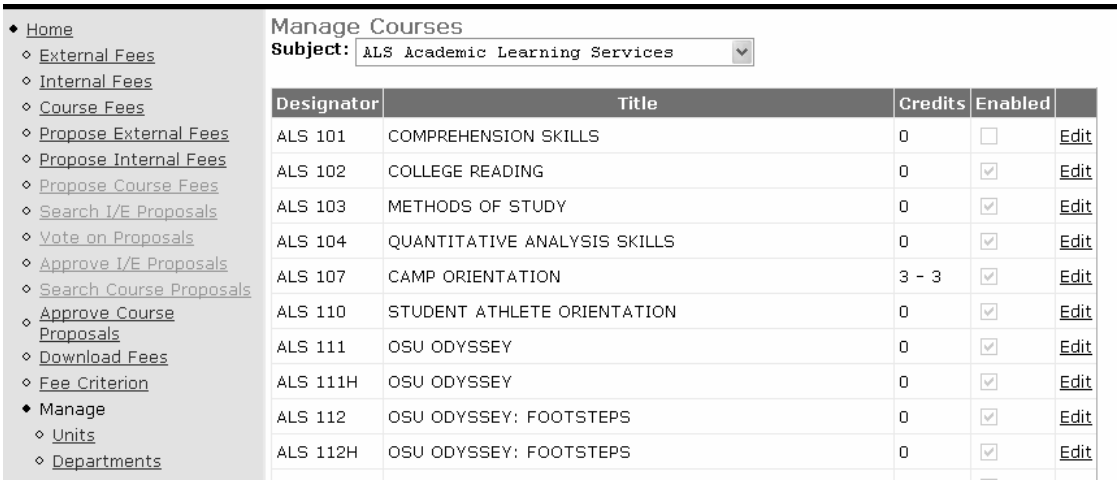

Figure 25: Manual Test 1

In a manual test, testers record each step and write down the result of every action. They have to write down their names and the time of testing as references. When other testers or developers check the page, they can see what they have done and what the problems are. If there are bugs, developers need to fix them and testers have to test it again. The Figure 26 below is the manual testing page for 'Manage courses'.

Manage Courses Test

This test will check the functionality of the page the administrator uses to enable and disable courses.

The target for this test is all the admin features available to the administrator for working with the course l enabling a course, and disabling a course.

Test Steps

Step No.	Step Description	Expected Result
1	Open the OSU Fees Site at http://osufeesstage.bus.oregonstate.edu and select the Login link. Login as yourself.	The Admin homepage is displayed with links to the admin tools on the left.
2	Select the Courses link under the Manage option from the navigation tree on the left.	The Manage Courses page is displayed with the first subject in the subjects drop-down selected and the courses for that subject displayed in a grid if any.
3	Select the Academic Learning Services subject from the drop-down list.	The courses for this subject should be listed in the grid.
4	Select the Edit link next to the first course ALS 101	The Edit link is replaced with an update and cancel link and the Enabled column now has an editable check box.
5	Change the check box to be the opposite of what it is now for ALS 101 and click Cancel.	The Edit link returns and the Enabled column not displays a disabled check box that does not reflect any change (it stays that same as it was before).
6	Repeat step 4 and 5 only choose update at the end instead of Cancel.	Same as step 5 except the change is reflected in the Enabled column.

Revision History

Author	Change Description	Time/Date modified
Wu, Zhi	Original Version	11/21/2005

Figure 26: Manual Test 2

5.3 Implementing the unit test (White Box Test)

The 'NUnit' test is a kind of white box test. The purpose of the 'NUnit' test is to test a specific functionality to see if the functionality works properly. In some cases, white box testing is a very complex and time consuming process. For example, if there are many branches and loops, he/she may not test all of these routines. To test all of these loops and braches is an impossible task. Basically, what one does is to test the important branches and loops to get an overview of the quality of the component. In our particular case, since one calls methods such as Selecting, Deleting, Inserting and Updating most often, the testing will not be very complicated. In addition, with the support of the testing template of Visio Studio 2005, white box testing becomes easier. Let's take a look at an example of testing the 'Course Subject' class.

More specifically, we showed an example of how to test the Inserting, Selecting, Updating, Selecting by filter and Deleting methods. Before testing, we had to create a subject object and set some random values for the object such as 'Subject Code', 'Title', and 'Enable'. After this, we called the Inserting method in 'Course Subject' DA class. The method returned an integer. Then we used the 'Assert's AreEqual' method to test if the returned value was 1 or not. If it was 1, that meant the Inserting method worked properly; if it was not 1, there were problems with this method. The snapshot below in Figure 24 is the code.

```
[TestMethod()]
public void CourseSubjectDACRUDTest()
{
    CourseSubject s = new CourseSubject();
    s.Enabled = false;
    s.SubjectCode = "CS999";
    s.Title = "The LAST CS class ever";
    string CourseSubject.Title
    try
    {
        #region Test Insert method

        int inserted = CourseSubjectDA.Insert(s);
        Assert.AreEqual(1, inserted, "The Insert method didn't work as expected.");

        #endregion
```

Figure 27: Unit Test 1

For the Selecting method, we passed the 'CS999" as a parameter to the 'Course Subject' DA's Selecting method. The method returned a subject object which was expected to

have the values that we inserted before. Next we compared the returned object's subject code with the 'CS999'. We also compared the title with the 'the last CS class ever'. The snapshot below in figure 2 is t8he code.

```
#region Test Select method

CourseSubject toChange = CourseSubjectDA.Select("CS999");
Assert.AreEqual("CS999", toChange.SubjectCode, "Single CourseS
Assert.AreEqual("The LAST CS class ever", toChange.Title, "Sir
Assert.IsFalse(toChange.Enabled, "Single CourseSubject Select

#endregion
```

Figure 28: Unit Test 2

For the Updating method, we created a 'Course subject' object and set particular values for the 'Subject Code', 'Title' and 'Enabled'. Then we passed the object as a parameter to the Updating method to see if it would return the number 1. If it returned the number 1, this meant that it worked properly. Otherwise there were bugs in the method. Below is the snapshot of the code in Figure 29.

```
#region Test Update method

CourseSubject changeTo = new CourseSubject();
changeTo.SubjectCode = "CS1000";
changeTo.Title = "Thought we had you fooled before...";
changeTo.Enabled = false;

int updated = CourseSubjectDA.Update(changeTo);
Assert.AreEqual(1, updated, "Updated didn't work properly");

#endregion
```

Figure 29: Unit Test 3

For the Selecting by Filter method, we had to create a filter first and set the 'Enable' field to true. We wanted to test the filter to get a collection of course subjects and test if the course subjects' 'Enable' field was true or not. Figure 30 below is the code.

```
#region Test Select by filter method

CourseSubjectFilter filter = new CourseSubjectFilter();
filter.EnabledOnly = true;

CourseSubjectCollection coll = CourseSubjectDA.Select(filter);
Assert.IsTrue(coll.Count > 1, "Multiple CourseSubjects weren't returned.");
for (int x = 0; x < coll.Count; x++)
    Assert.IsTrue(coll[x].Enabled, "Returned a wrongly filtered CourseSubject.");

#endregion
```

Figure 30: Unit Test 4

For the Deleting method, we passed a particular 'Subject Code' such as 'CS1000' to the Deleting method. This method would return back an integer value. If this value returned by the method was equal to 1, it passed the test. Otherwise it was failed. Figure 31 below is the code.

```
finally
{
    #region Test Delete method

    int deleted = CourseSubjectDA.Delete("CS1000");
    Assert.AreEqual(1, deleted, "The Delete method didn't work as expected.");

    #endregion

}
```

Figure 31: Unit Test 5

In the project, we did the black box test, i.e., the manual test in the application. We also did the white box test. For example, we tested the Inserting, Deleting, Updating, and selecting methods. This chapter summarizes how to test an application. Because these methods return the correct values, we have more confidence in our code. Our effort will also reduce the chance of a software disaster and software maintenance fees in the future.

CHAPTER 6
Conclusions

The OSU Fee Tracking project has provided the staff members with a convenient and standard way to manage fees. The fees are created and monitored by a reasonable process. Excepting administrators, all other people such as committee members and originators cannot create a fee only by themselves. The whole process begins with the originator, and then many committee members vote for a fee. After this, administrators check the status of a proposed fee and make a decision to approve it or not. If problems happen with the fee created, we can check the process and find out reasons.

For the students, if they have confusion with fees, they can just go to the web site to check or download the fee book. In this way, students do not need to call the staff members to ask for information about fees. This reduces staff members' work load, and students can also get the information without delay.

In the first chapter, we represented the big picture of the system and demonstrated how the application worked. Next we analyzed the requirement of the system before we designed it. Then we talked about the architecture of this system especially for the MVC pattern. After this, we talked about how to implement this system based on the requirement and architecture. Finally, we explored how to test functionalities using black box and white box testing framework in .Net platform.
In the future, we would like to add a reporting service to analyze the history data and provide administrators with help in making better decisions about fee policy.

References

[1] O'Reilly Programming C#, 4Ed : http://safari.oreilly.com

[2] Wrox Press Professional C# 3rd: http://www.wiley.com

[3] O'Reilly Programming ASP.Net 2nd Edition: http://safari.oreilly.com

[4] O'Reilly ASP.Net in a nut shell: http://safari.oreilly.com

[5] Addison Wesley essential ADO.net: www.aw.com/cseng/

[6]A Programmer's Guide to ADO.net: http://www.apress.com

[7].Net Framework Essentials: http://safari.oreilly.com

[8] http://www.15seconds.com/issue/040922.htm

[9] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides,
Design Patterns: Elements of Reusable Object-Oriented Software