Financial Aid Data Warehouse

A Manuscript

Submitted to

the Department of Computer Science

and the Faculty of the

University of Wisconsin-La Crosse

La Crosse, Wisconsin

by

Yi Qian

in Partial Fulfillment of the

Requirements for the Degree of

Master of Software Engineering

May, 2008

Financial Aid Data Warehouse

By Yi Qian

We recommend acceptance of this manuscript in partial fulfillment of this candidate's requirements for the degree of Master of Software Engineering in Computer Science. The candidate has completed the oral examination requirement of the capstone project for the degree.

_____	_____
Dr. Thomas Gendreau Examination Committee Chairperson	Date
_____	_____
Dr. Kasi Periyasamy Examination Committee Member	Date
_____	_____
Dr. Mao Zheng Examination Committee Member	Date

ACKNOWLEDGEMENTS

First, I would like to thank my project advisors Dr. Thomas Gendreau and Dr. Kasi Pariyasamy for their valuable advises. All the concepts from previous courses are just vague ideas, because their comments and guidance, I started to realize those concepts in a real world project. Without them, this project and dissertation would not have been possible.

Second, I would like to thank Mr. Josh Anderson (Previous ITS Developer) and Mr. Michael Taylor (Database Administrator, ITS). Their help are just tremendous. Especially Josh Anderson, he initiated this project and provided valuable knowledge about the school database.

Third, I would like to thank my project sponsor Ms. Teri Thrill, Ms. Louise Janke and Ms. Janis Van Ruden for spending their valuable time to discuss the project every week. It is a pleasure to work with a group of brilliant women.

I also would like to thank the Computer Science Department and the University of Wisconsin – La Crosse for giving me the opportunity to pursue a high degree.

Finally, I would express my thanks for my family, without their support, I would not be able to achieve this far.

ABSTRACT

QIAN, YI, “Financial Aid Office Data Warehouse”, Master of Software Engineering, May, 2008, (Dr. Thomas Gendreau and Dr. Kasi Periyasamy).

Recently, the institutional research office has increasingly emphasized campus decision applications in which current and historical data is comprehesively analyzed and explored in order to support high-level decision making. Many characteristics of decision support queries make the current database system inadequate: The query clause required by institutional research office often contains many AND and OR conditions. Particularly OR conditions are poorly handled in the current school database systems.

The institutional research office often needs to pose several related queries. Since there is no convenient way to express these commonly occuring families of queries, ITS has to write them as a collection of independent queries, which can be tedious. Further, the DBMS has no way to recognize and exploit optimization opportunities arising from excuting many related queries together. Because many of the analyses performed are recurrent and predictable. Data warehouse provides access to data for complex analysis, knowledge discovery and decision making. This report describes the development of financial aid data warehouse, especially the activities performed in each stages, data warehouse design and data dictionary, the challenges encountered, issues, current status of the project, limitation and possible improvement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY

ANT

Another Neat Tool. A popular Java build tool

API

Application Program Interface. The interface through which an application accesses the operating systems and other services.

APACHE LOG4J

Logging framework for printing log output to different local and remote destinations. Log event can be selected and filtered at run time.

APACHE POI

Poor Obfuscation Implementation. APIs for manipulating various file formats based upon Microsoft's OLE 2 Compound Document format using pure Java.

APACHE STRUTS

A web application framework simplifies the building of web applications based on MVC design pattern.

APACHE TILES

A template framework simplifies the development of web application user interface.

BRIO

A database query tool provides Graphic User Interface in which the user can construct SQL query by drag and drop tables, fields of the database without SQL knowledge.

CSS

Cascading Style Sheet is a style sheet language used to describe the presentation of a document written in a markup language. It is most commonly used in HTML and XHTML, but also can be applied to any XML document.

DTD

Document Type Definition. It defines the legal building blocks of an XML document. A DTD can be declared inline inside an XML document, or as an external reference.

EAR

Enterprise ARchive is a file format used by Java Enterprise Edition for packaging one or more nodules into a single archive so the deployment of various modules onto an application server can be simultaneously and coherently. It contains XML files called deployment descriptors which describe how to deploy the modules on an application server.

HTML

HyperText Markup Language. A markup language used for crating web page contents.

IEEE

Institute of Electrical and Electronics Engineers. An international organization whose constitution describes their purposes as “scientific and educational, directed toward the advancement of the theory and practice of several engineering fields including computer science.

ITS

Information Technology Service. The department provides service and support to school computer systems

JNDI

Java Naming and Directory Interface. A part of the Java platform provides applications based on Java technology with a unified interface to multiple naming and directory services. You can build powerful and portable directory-enabled applications using this industry standard.

JSP

Java Server Page is a Java technology that is used to develop dynamic web pages.

SMTP

Simple Mail Transfer Protocol is a standard protocol to send email across a network.

SQL

Structured Query Language. A popular database query language.

SRS

Software Requirement Specification. A document format supplied by the IEEE for specifying the requirements of a software system.

SSL

Secure Socket Layer is a cryptographic protocol that provides secure communication on a network over a socket connection.

UML

Unified Modeling Language is an industry standard for specifying, visualizing, constructing and documenting the software system.

WAR

Web ARchive. A Jar file used in web applications.

XML

eXtensible markup Language. A W3C recommendation for creating special-purpose mark-up language and is widely used for exchange data, store configuration and other usage.

1. BACKGROUND INFORMATION

The University of Wisconsin-La Crosse currently uses a legacy database system called Campus Administrative System (CAS) which was developed and maintained by Unisys for a long time. This legacy database system is not a relational database and doesn't conform to SQL92 standard. When the university migrated from Unisys to Oracle few years ago, the legacy Unisys database was moved to new Oracle relational database system without modification. One of the clients of this legacy database is the institutional research office which often uses ad hoc queries and generates reports in order to analyze student data, predict the future students' interest and adjust the university policy corresponding to the changes. To perform trend analysis, the institutional research office is interested in both current student data and historical student data.

The financial aid office uses ad hoc queries to handle students' financial needs. There are two types of financial assistances – need-based and non-need-based. The great variety of criteria required for the financial aid award makes it difficult to use a single query to determine the eligibility, each type of the financial assistance needs its own query. The financial aid office wants to have an application to construct ad hoc queries without SQL knowledge. Currently, financial aid office has to ask an ITS programmer to construct all the queries and many queries are just slightly different. The whole procedure is error-prone, any miscommunication can cause an error, because the query string verification requires SQL knowledge, financial aid office is unable to ensure the correctness of the query string. Because of the increasing amount of querying, the financial aid office wanted to develop a software system that has a simple user interface and will allow them to construct ad hoc queries without any SQL knowledge.

The data required for both offices doing ad hoc queries are located at different databases and even in the spreadsheets. Fairly large amount of data are not residing in

the CAS database system. Because these databases are managed by each office, in order to perform tasks independently, many data fields in the CAS database system have to be duplicated in these databases. Those data not only need extra hard drive space to store, but can easily become outdated or out-of-sync with the CAS database system as well.

Even for the CAS database, the following problems make it inadequate to perform ad hoc queries.

- Poor data quality

  The data are entered by humans, the mistakes such as typing errors during the input are inevitable. Due to the lack of basic quality assurance, such as consistency check, after years of accumulation of mistakes, the data quality becomes very poor. But data accuracy is critical to the financial aid office when the financial aid office tries to allocate the limited financial aid resources to match the students' needs. Because there is no synchronization for the same data fields between the CAS database and other databases, the financial aid office has to run many unnecessary queries to guarantee the data is up-to-date and accurate and manually synchronize all the data fields.

- Non-relational database design

  The CAS database is migrated from a non-relational database to a relational database without modification. This makes it difficult to generate ad hoc queries. Sometimes, intermediate views or tables have to be created and joined in order to perform the complex ad hoc queries. Unnecessary table creation and join also increases work load on the current servers as well. It would strain the capacity of the servers and decrease response time for both query users and the operational programs. Frequent modifications such as adding new tables, removing tables and changing table structures make the situation worse. There are three major defects in the CAS database design.

  1. Tables are not normalized.

     Duplicated data fields exist throughout the tables. Duplicated data fields cause consistency problems, which cannot be checked internally in the database

system and make it hard to decide which table to query and to perform the query optimization.

2. No relationships among the tables.

   No relationships are represented in the CAS database making it difficult to join tables during complex ad hoc queries. Intermediate views and tables have to be created.

3. Non-Modularized Implementation.

   The CAS database system is an old legacy system, the system implementations were tied up with the underlying data. The database changes require the changes in the data transaction processing. New source fields must be identified and data transaction programming may need to be altered to reflect changes.

Reports are very difficult to generate with only SQL statement. For example, The Princeton Survey conducted by US News and World Report, based on which the university is ranked is highly complicated. To generate reports for Princeton Survey requires the creation of many temporary views and complex queries run against the views. It requires ITS programmer involvement and puts extra work load on already short-handed ITS. Currently 45 questions need 25 page long queries to generate the results.

ITS intended to provide a data warehouse test environment for institutional research office to practice the decision-driven application model. But because the university database system is migrating from enterprise to PeopleSoft, ITS currently does not have the resources to implement a data warehouse test environment, ITS wants this project to provide the data warehouse test environment to the institutional research office.

This capstone project describes the activities of developing a simple data ware house for the financial aid office; typical activities include gathering the requirements, design and implementation. The financial aid data warehouse supports ad hoc queries and generates reports for both the institutional research office and the financial aid office. It will have a centralized location for the data, a graphic user

interface to allow the financial aid office to create ad hoc queries without any SQL knowledge and automatically generate reports without ITS programmer involvement.

2. BRIEF INTRODUCTION OF DATA WAREHOUSE

In modern organizations, users of data are often completely removed from the data sources. Many people only need read-access to data, but still need a rapid access to large volumes of data. Such data often comes from multiple databases. Organizations have increasingly emphasized applications in which current and historical data is comprehensively analyzed and explored in order to support high-level decision making. [5]

Because many of the analyses performed are recurrent and predictable. Data warehouses provide access to data for complex analysis, knowledge discovery and decision making.

According to Alex Berson and Stephen J. Smith a data warehouse is a database designed for analytical tasks using data from multiple applications [2] with the following characteristics:

- Relatively small number of users with relatively large amount of data
- Read-intensive
- Updated periodically
- Contains current and historical data
- Contains a few large tables
- Supports ad hoc, unstructured, heuristic queries

A data warehouse includes historical data and is refreshed according to careful choice of refresh policy, usually incremental. Data warehouse update is handled by the data warehouse acquisition component that provides all required preprocessing. Depending on the size of data, data warehouses are divided into enterprise-wide data warehouses and division focused data marts. Data warehouses are enterprise wide, while data mart is generally targets on a subset of the organization, such as a department and are more tightly focused. The data scope is limited to the focused target.

A significant issue in data warehousing is the quality control and the consistency of data. Although the data passes through a cleaning function during acquisition, quality and consistency remain significant issue for the Database Administrator. The quality control and consistency of data includes naming and domain definitions.

The following table lists some of the differences between a data warehouse and a transactional database:

	Transactional Database	**Data warehouse**
Data Content	Current	Current and historical
Data Organization	By Application	By Subject
Data Stability	Dynamic, reflect the real-time changes	Non-volatile. Changes are far less and may be regarded as non-real-time with periodic updating
Data Structure	Optimized for transactions	Optimized for complex queries
Access Type	All Kinds include Read/write	Mostly only read and aggregated add to

Table 1. Data Warehouse Comparison

3. SOFTWARE DEVELOPMENT MODELS USED IN THE PROJECT

The two most common software development models are waterfall model and prototype model.

Waterfall Model

The waterfall model also called the Classic Model or the Sequential Model is the most well understood software development model. The waterfall model breaks the software development into different sequential stages – Requirement stage, Specification stage, Design stage, Implementation stage, Testing and Integration stage and Maintenance stage. The waterfall model works better when requirements are easily to be identified.

The advantages of waterfall model are the following:

- It is documentation-driven. Documentation is produced at every stage.
- Testing is inherent in every phase, continuously as well as at end of each phases
- It controls schedule, budget and documentation and milestones can be set clearly

One of disadvantages of the waterfall model is it assumes that requirements are set, stable, and fully evolved before analysis begins, because development progresses linearly through the phases from requirements through system deployment. A phase is revisited only if artifacts created in that phase fail inspection, review, or test. Because the users interact with the development team infrequently after the requirements have been specified, any mistake or misunderstanding in the requirements analysis stage takes much longer time to be reflected at the test stage.

Prototyping Model

The modern software systems are so much closer to the user that their voices cannot be ignored; they'll reject the system if it doesn't meet their needs. [14]

The reality of modern software development is that change is unavoidable and must therefore be explicitly accommodated in the software development life cycle. It is not an error that must be fixed; it's a natural aspect of system construction. The change is not isolated to requirements, but the requirements example is the most immediate and most significant.

The prototype model is a cyclic version of the waterfall model. In this model, the development is in a loop till the final product is delivered. After each prototype is created, it is given to the customers for evaluation. The development team uses customers' feedback to refine the product. The final software will be ready after a finite number of iterations. In this model, the software evolves as a result of periodic communication between the customer and developer. This is the most popular development model in the contemporary industry, as it is very difficult to comprehend all the requirements of a customer in one shot. [14]

The advantages of prototype model are the following:

- Incomplete, incorrect, inconsistent or unfeasible requirements may be identified during the prototype iteration.
- Misunderstanding and miscommunication may be reduced
- Missing functionality may be detected earlier
- Requirement and design can be refined

The disadvantages of prototype model are the following:

- Project may fall into an infinite loop of iteration and never end due to the requirements creep.
- Prototype is often used as final production. This is not an issue in other engineering disciplines, since the prototype could not be used in the final systems. In prototypes it is common to defer structural and architectural

concerns and to give scant consideration to fundamental practices such as exception handling. [14]

At the beginning of the financial aid data warehouse project, the waterfall model was chosen to prevent the requirements creep. Many software development projects encountered the problem of requirements keeping changing which caused project size to expend and the project completion delayed. Intensive interaction increases the functionality desired by the customers. The problem of the waterfall model is that all requirements must be specified in advance. Unfortunately, because I lacked of knowledge of the problem domain, the complication of the problem domain was greatly underestimated when the project started. In the meantime, customers were unclear about the requirements of the application and what functionalities they want to be implemented. Requirements grew and changed throughout the process and beyond, calling for considerable feedback and iterative consultation. As the project went on, I realized that if I did not adapt the solution to these changes, the costs of accommodating such requirements would escalate exponentially. Finally, I decided to abandon the waterfall model and started using the prototype model.

4. DEVELOPMENT OF FINANCIAL OFFICE DATA WAREHOUSE

In a normal prototype model based development, the user interface is developed first to give the customer an idea of what the project is going to look like and what functionalities the project is going to have. In this project, the normal prototype model development procedure was reversed because of some customers have in-depth knowledge of using querying tool Brio to query the database and they wanted to have data warehouse developed first. The data warehouse was designed, development and tested first. The user interface was designed only after the data warehouse was completed.

4.1 System Architecture

Because the project is going to be a web application, it is developed as 3 + 1-tier system.

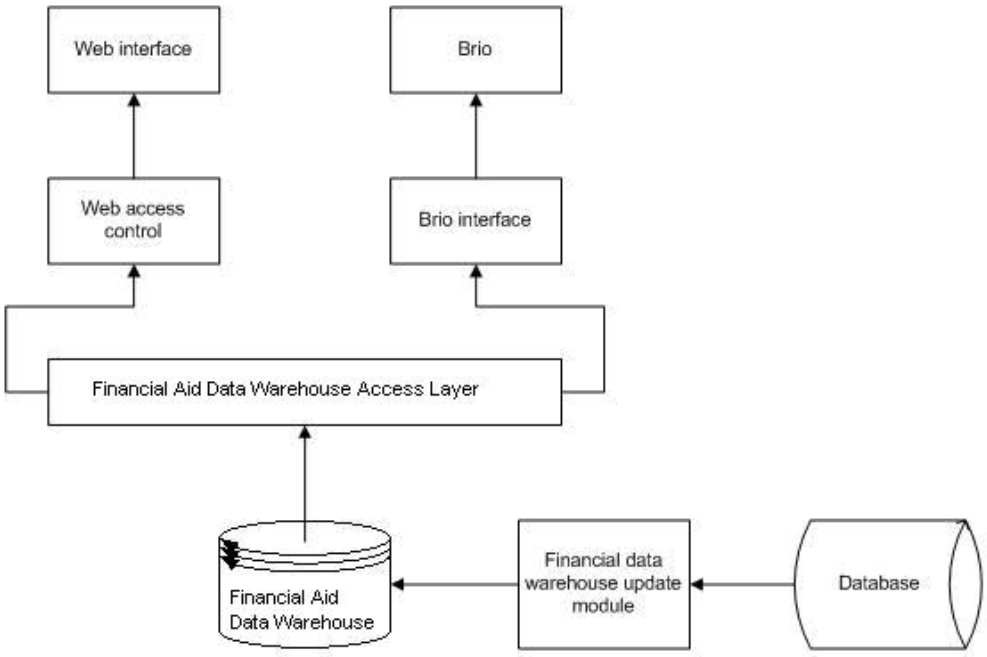

Figure 1. System Architecture

Following are the detailed explanation of each part of the figure 1.

- The database is a symbol of all the data sources required by financial aid data warehouse including the CAS database, admission database, spreadsheet file and paper works.
- Financial aid data warehouse update module is the data acquisition layer which retrieves the data from various data sources, cleans the data and stores the data into financial aid data warehouse. It is a totally separated from other implementation.
- Financial aid data warehouse access layer manages the data warehouse and handles all the queries. It is the database tier in the 3-tier structure.
- Web access controller and Brio interface layer handles client requests and redirect user request to the appropriate under layer classes. It is the business tier in the 3-tier structure.
- The web user interface is the presentation tier.

Data can only flow from the CAS database to the financial aid data warehouse. No method will be implemented to flash data from the financial aid data warehouse back to the CAS database in the project.

Although the data warehouse management methods were implemented and tested in the project, they are currently disabled waiting for a decision by institutional research office about how the financial aid data warehouse update will be done. So currently the financial aid data warehouse cannot be updated through web user interface.

4.2 Requirement Analysis and Specification

The financial aid data warehouse requirements include the following:

- Identify all the data fields related to the student financial aid

  There are thousands of data fields existing in the school database systems and only small portion of data is related to the student financial aid.
- Identify the data sources for the financial aid data

  There are hundreds of tables existing in the school database systems and because non-normalization, many data fields are duplicated over multiple tables.
- Extract from multiple, heterogeneous data sources

  The data required for the financial aid data warehouse reside in many resources.
  - The CAS database contains most portions of the data.
  - The newly developed admission database includes a table of students' payments.
  - The spreadsheet files reside on financial aid office PC.
  - Some data such as graduate student classification code exists on paper only.

The fundamental requirement is to create the financial aid data warehouse from all the resources. The following steps are necessary to meet this requirement.

1. Create a centralized view for the tables from the different databases and generate code to automatically transfer the data into the financial aid data warehouse.

2. Move the data files into a centralized location which is the development machine and generate code to automatically transform data from files into financial aid data warehouse.
3. Manually enter the data that exists on paper into the tables.

- Inconsistent and incorrect data

  During the development, I discovered some inconsistent and incorrect data. For example, in the CAS database, there exists student record whose high school class size is over 8000. Another example is every year thousands of student financial award amount records are different in the tables which are most important to the financial aid office. The following steps were taken to handle the inconsistent and incorrect data.

  1. Clean data by generating code to automatically filter out the incorrect and inconsistent data for the financial aid data warehouse based on criteria provided by the institutional research office and the financial aid office.
  2. For the same field existing in different tables, generate the code to compare the value and record differences
  3. Save the inconsistent data and incorrect data in a spreadsheet for future analysis.
  4. Email the spreadsheet to the institutional research office and the financial aid office to allow easy and conveniently access to the result.

- Data warehouse normalization

  Data warehouse is usually not normalized in practical, but the financial aid data warehouse will be normalized upon the institutional research office request.

- Relational database implementation

  The CAS database is not a relational database, which makes complex ad hoc queries difficult to construct. By designing financial aid data warehouse in the relational database system will improve the abilities of constructing the complex ad hoc queries, reduce or even eliminate the construction of the temporary tables.

- Historical Data

  The CAS database does not have any students' financial historical data. Old data has been overwritten by the updates. But the financial aid office wanted to keep

all the historical data for future analysis and decision making. The financial aid data warehouse will keep all the students' financial historical data and implement the functionality which allows the authorized user to purge the data.

- Security

  Because the sensitive of the data this application deals with, both financial aid office and ITS want maximum security protection. The security was enforced by the following implementation.

  6. The network connection is using the secure socket layer connection – 'https' instead of 'http' to protect from unauthorized access during network traffic.
  6. Every data field is encrypted in the financial aid data warehouse.
  6. The application times out at client side after predefined period of inactivity.
  6. The user has to be authenticated in order to use this application.
  6. The application logs every activity performed by the logged-on-user
  6. Every data warehouse operation will be validated, any user SQL query trying to alter the financial aid data warehouse structure or data such as *INSERT, DELETE, DROP,* etc. will be dropped and the whole application automatically logs out.

4.3 User Interface

Both the institutional research office and the financial aid office are not willing to install any more applications onto their desktops and both offices want the software can be maintained remotely without involving changes of their PCs. In this case, web-based application is the only solution. Another reason to chose web-application is that this project is developed under ITS supervision and ITS requested this project should be developed as a web application.

Pros of web application

- Upgrade

  The ability to update and maintain applications without distributing and installing software on potentially thousands of client computers is a key reason for the popularity of the web applications.
- Operating System Independent

  Web applications support standard browser features should have exactly same user interfaces and functionalities regardless of the operating system running on the client side.
- Remote Access

  Web applications can be accessed from anywhere the Internet is available

Cons of web application

- Security

  A poorly designed web application can impose great security risk onto the application and the data it deals with.
- Connectivity

  Web applications rely on persistent and unmanaged connectivity. If the connection is unavailable, then no web applications can be available.
- Too Many Technologies

  A standalone application may require using several different technologies. But web-application definitely needs integrating lot more technologies. For example, if the application developed in Java, then minimum technologies required are Java, JSP, HTML, CSS, JavaScript, XML, if using framework, it requires even more technologies. Many times the developers find out they are actually trying to solve the problem between the technologies rather than coding.
- Hard to Debug And Test

  The web-application is so hard to debug, the developers cannot step through any web pages. It is impossible to insert break points to any of web pages and it does not have any stack trace messages to look into. The frameworks only

provide a set of functions which have been thoroughly tested, but the customized presentation layer code debugging and testing are still painful.

4.6 User Classification and Characteristic

Upon the customer request, the application has been designed to have three types of users – super user, general user and operator. All the functions implemented on the user interface are available to the super users. General user can do queries, but no user account management functions available to the general users and only update function is available to operator(s).

User Catalog	Functions
Super user	All functions, including user management to add or remove users
General user	All the queries. No user management
Operator	Only one function: Update

Table 2. User Catalog

5. DETAILED DESIGN

5.1 Data Warehouse Design

The financial aid data warehouse used snowflake schema by combining two star schemas – term fact schema and financial aid schema.

One of the most important designs of the financial aid data warehouse is the data warehouse acquisition module. It includes extracting data from all the data sources, cleaning the data and storing the data in the data warehouse. Cleaning data is much more important than extracting data and loading data because the data quality. The functionalities of cleaning data including:

- Convert to common data names and definitions
- Establish default values for missing data
- Compare same data set from different data sources
- Identify and record both incorrect and inconsistent data based on the criteria provided by the customers.
- Email super user the recorded data for further investigation.

Another important design of the financial aid data warehouse is the data warehouse management which includes:

- Update financial office data warehouse
- Audit and reporting financial aid data warehouse usage and status
- Purge data
- Security and priority management

5.1.1 Entity Relation Diagram

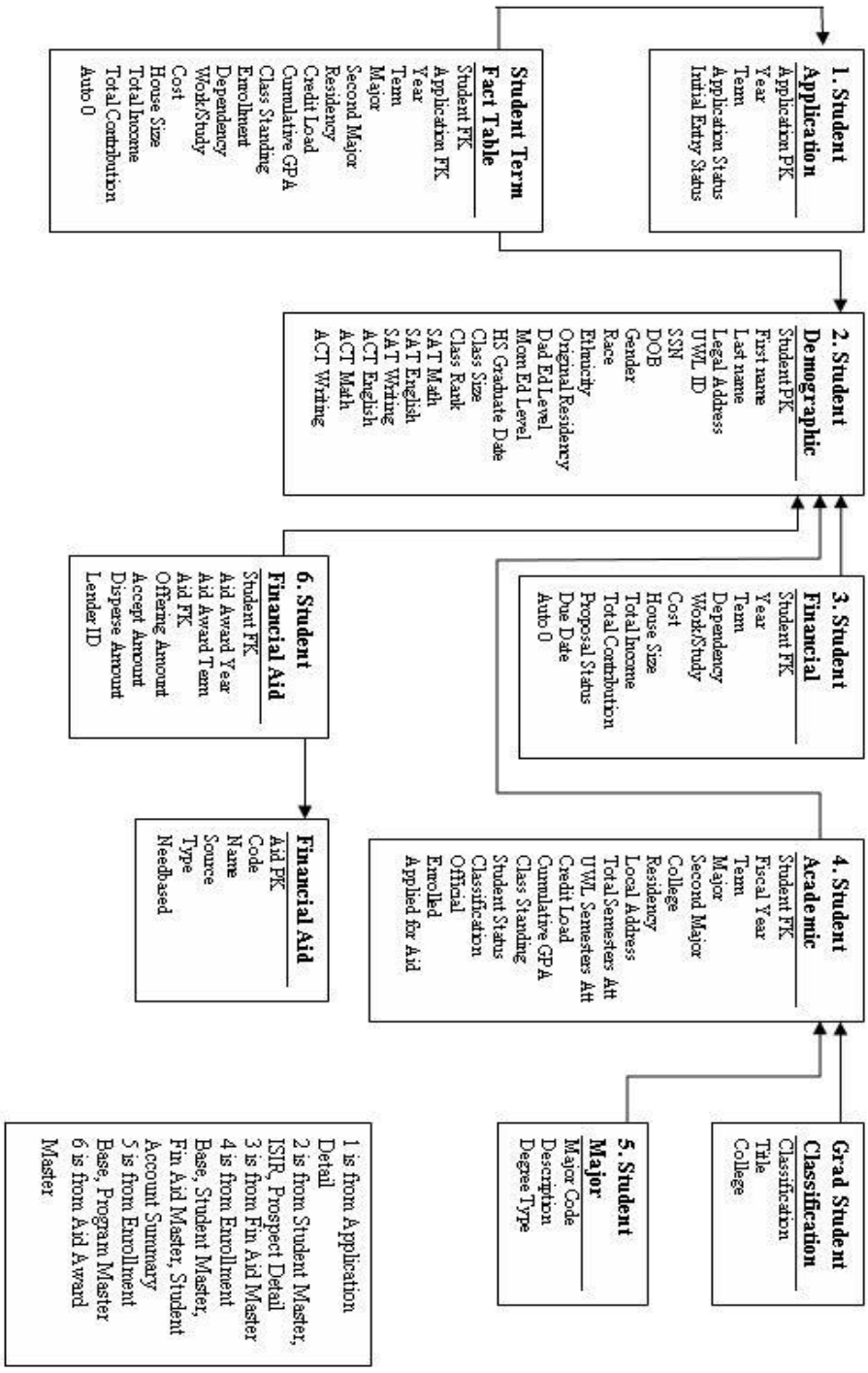

Figure 2. Entity Relation Diagram

The financial aid data warehouse includes nine tables. The Student_Term_Fact table and the Financial_Aid_Fact table are called “fact table” in term of data warehouse terminology and the rest of the tables are “dimension tables”. Each table has unique primary key generated automatically and has a foreign key related to another table, please refer to the figure 2 for the relations between the tables.

Because the financial aid data warehouse will also be a test environment for the institutional research office to practice decision-driven application model, although the normalization is not recommended to the data warehouse, due to the institutional research office request, the financial aid data warehouse is normalized. The normalization makes it difficult to construct some complex ad hoc queries because the extra tables join. A Java class is devoted to those small queries and complete the intermediate processing.

5.1.2 Improvement over the CAS Database

Because the financial aid data warehouse is a totally new design, I try to make as many improvements as possible. The improvements come from five different aspects.

1. Data readability

   In the CAS database, most values are numeric. People who are not familiar with the data set have no idea what the data values are. For example, semesters are listed as 1, 4, 7 and 9. In the financial aid data warehouse, semesters are represented as “summer”, “fall”, “j-term” and “spring”.

2. Separate the aggregated field into multiple fields.

   For example the address field in the CAS database is represented as one field separated by space or tab. This design and implementation in the CAS database makes it very difficult for other applications to retrieve partial information, such as state information. In the financial aid data warehouse, address information is stored in separated fields for street, apartment, city, state and zip.

3. Create a separate primary key instead of using student social security number

In the CAS database, because it was migrated from a non-relational database, neither primary key nor foreign key is explicit defined. The student social security number is included in every table used as primary keys. The major problem of using social security number as primary key is for foreign students because they do not have social security number when they enrolled in the university, the database administrator has to assign a temporary unique number to social security number field of the student record and the database administrator has to change it in every table after the student obtained the social security number from social security office. This process is time-consuming and error-prone. In the financial aid data warehouse, first, social security number is designed as a field of a table. Changes made to this field will not affect other tables. Second, primary keys are automated generated by the application and hence human error can be avoided.

4. Create relationships between the tables

   Because no primary keys or foreign keys are defined in the CAS database, the relationships are not represented in the CAS database. This leads to inconsistent data and the problems with joining tables. Each table in the financial aid data warehouse has foreign key to connect to other table(s).

5. Automated synchronization procedure

   The two most important tables AidAwardMaster and StudentBillingDetail for the financial aid office currently reside on different databases. The AidAwardMaster table is suppose to have all the student financial aid data, but many students don't report their scholarships or grants received from outside resources, therefore the AidAwardMaster table has to be updated periodically. The StudentBillingDetail table contains students' tuition and billing information and is recorded outside financial aid office. Because these two tables are residing in different databases, consistency checks cannot be performed. The financial aid office has to manually synchronize the AidAwardMaster table with the StudentBillingDetail table and because the financial aid office has no authorization to access the StudentBillingDetail table, financial aid office has to request the payment data from cashier's office

and then manually query each data for the comparison. After comparing data, modifications to the AidAwardMaster are made by manually running the SQL update. The whole synchronization process is complicated, time-consuming and error-prone. In the financial aid data warehouse, this process has been automated. There is a web page to allow the financial aid office to run the comparison queries against those 2 tables to find out the differences. Any financial aid award only existing in the StudentBillingDetail table will be automatically added to financial aid data warehouse. The program will generate a spreadsheet for the records that have different amounts or types in those two tables. The financial aid office can look at the data in the spreadsheet file. After the financial aid office has reviewed and made a decision, another web page allows the financial aid office to upload the spreadsheet file to the server and the application will automatically update to add these reviewed data into financial aid data warehouse by uploading completed.

5.1.3 Decision Tree

Every year, the institutional research office needs to handle many reports. Those reports are used for analysis purpose. They can be represented in a decision tree format. The following figure is a decision tree model of the Princeton Survey conducted by the US New and World Report.

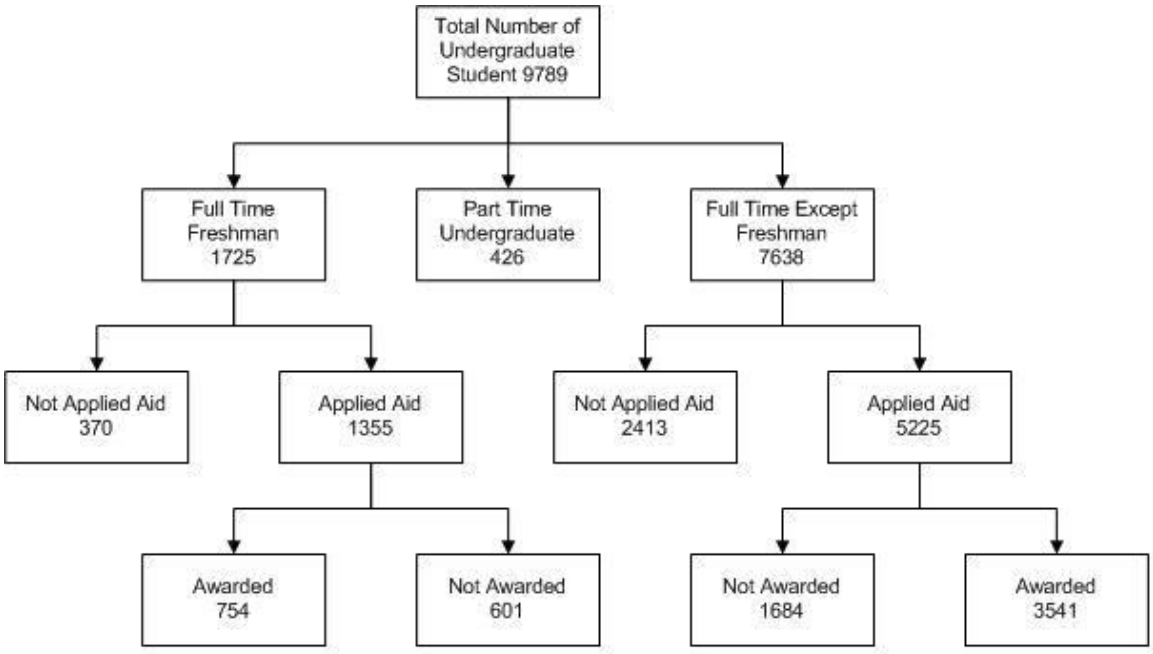

Figure 3. Decision Tree for Princeton Survey

A decision tree is a resulting model presented in a tree structure used for both predictive and descriptive purposes. The decision tree is a very popular data mining technique. Decision trees are most commonly used for classification. [12] The decision tree grows from the top node, which is called the root node. The root node for this decision tree given in figure 3 represents the total number of undergraduate students and starts growing upside down, *splitting* the data at each level to form new nodes. The intermediate nodes are called *branches*. Nodes that are at the end of branches are called *leaf* nodes.

The decision tree provides interesting descriptive information about the data. There are often additional interesting and potentially useful observations about the data that can be made after a tree has been induced. For example, figure 3 clearly shows that new freshman have no advantage over other students in receiving financial aids.

A Java class is devoted to handle the decision-tree-growing by calling the SQL queries and computing the final results to construct each node or leaf.

5.2 Classes

5.2.1 UML Class Diagram

UML is widely used in the software design to help identifying the problem domain and implementation. Because the class diagram is not at absolutely abstract level, it gives the developers a broad view of implementation and because the class diagram is still at the design stage, changes can be made at relative low cost.

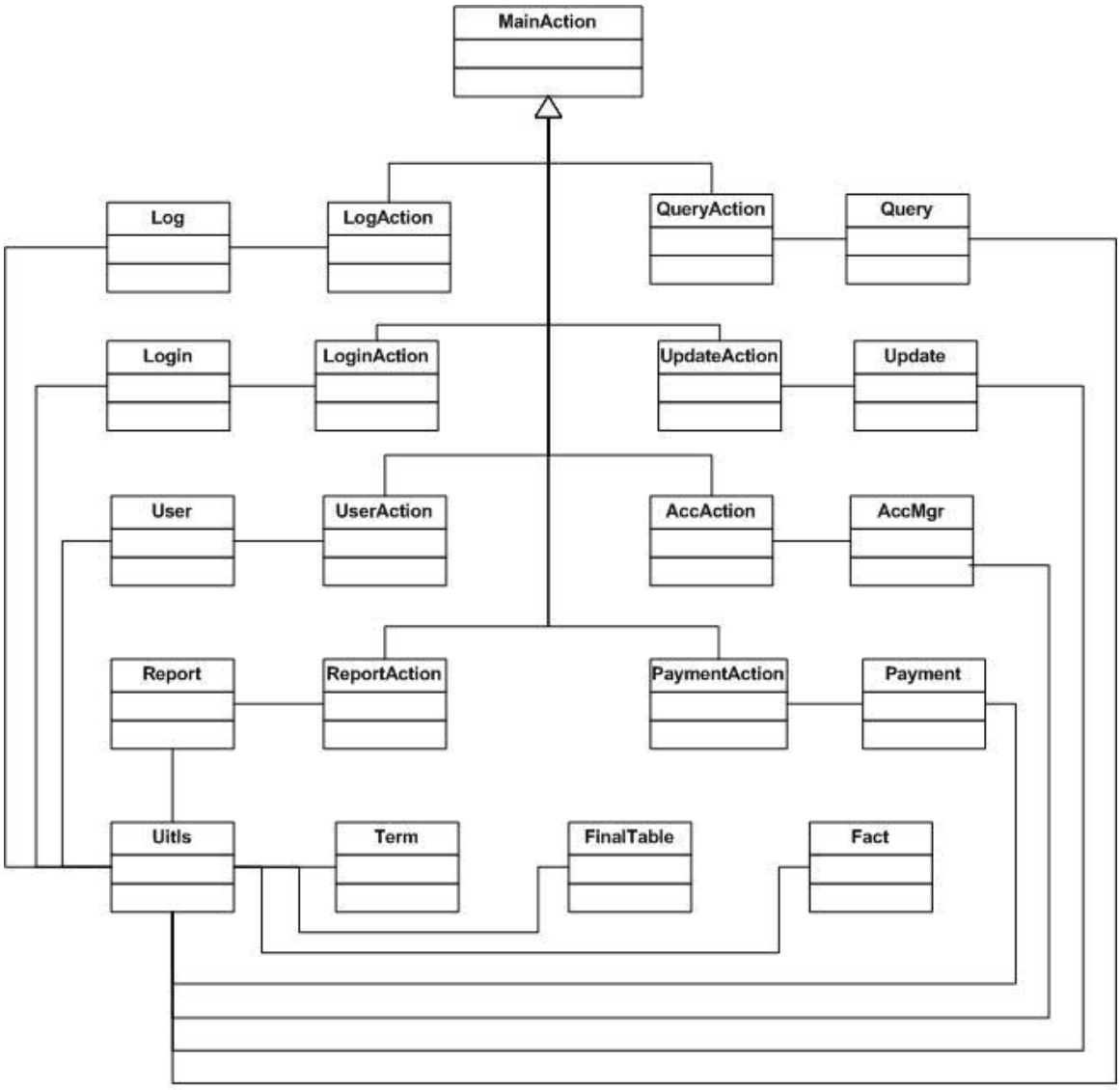

Figure 4. UML Class Diagram

The classes are divided into three modules.

- Data Warehouse Module

  This module consists of Utils, Term, FinalTable and Fact four classes and handles table populating, data cleaning and table updating. For each query request, it will validate the query, create the database connection, execute the query and return the result.
- User Interface Module

  This module consists of all the nine action classes. It handles the page flow based on the user selections, passes the user input to the controller module and passes the result from controller module to JSP page for displaying.
- Controller Module

  This module consists of eight classes – Log, Login, User, Report, Payment, AccMgr, Update and Query. It connects the user interface module with data warehouse module and controls the page flow. Each class handles the corresponding action class; receives user request, applies business logic upon the user request, requests the data from the data warehouse module and returns the results back to the action class.

5.3 Implementation

5.3.1 Development Environment

ITS is one of the customers for this project and all the ITS developed web applications are implemented in Java with the Struts framework. Because the project has been developed as a web-based application, ITS required that the project should be developed using Struts.

5.3.2 Implementation Challenges

The following challenges were encountered during the implementation.

1. Framework learning curve – The learning curve is high, especially because of there is a little local expertise and because of the poor documentation. The

frameworks used in the project are struts framework, tiles framework for the page display, validator framework for user input validation, log4j framework for logging and POI framework for Microsoft office file handling. Learning the frameworks takes time but also has the following benefits:

- Frameworks provide a series of tools and components to build applications with to increase productivity of the developer. Otherwise those features may require months of planning and development.
- Frameworks help to encode best practices because the development teams are always trying to use the best approaches.
- Another benefit is that it allows the application to be highly platform independent, this is true at least for the Struts frameworks. The application can run on any web server and operating system without modification.
- It allows developers to think about (and design) complex applications as a series of relatively simple Model-View-Controller (MVC) components. This leads to better, more consistent, and more easily maintainable designs.
- In addition, it helps avoid the common pitfall of having each developer on a project choose a different approach for their works.

2. Debugging – Web applications are harder to debug than stand alone applications, it takes much more time to debug web applications and no good tools are available. In order to make debugging easier, I developed a debugging page to print out any content in the Struts container called ComponentContext and any content in the HttpSession. It was included in every JSP page and when the page loads, it dumps everything in the content to the browser. It allows me to check whether the variables have correct values to determine the problem and it was turned off in the production environment.

   No version control – No version control makes it difficult to roll back to the safe stage and difficult to merge the development. For the rollback problem, I always keep an up-to-date workable copy, and adding new code to the existing project.

For merging, I use a free comparison software to compare two copies, found the differences and manually merged differences together.

5.3.3 Concurrency issue

The concurrency issue in this project is that data warehouse might be updating while users are querying data. These may cause incorrect or inconsistent result. Data consistency is handled in following two levels.

- At the Database level, the transaction isolation level is set to Serializable, the most restrictive isolation level. It prevents other users from updating until the current transaction is completed.
- In the code level, the auto commit is set to false and it is explicitly coded commit method at the end, when the all the transactions completed without error, before the program exits.

5.3.4 Decoupling

The benefits of decoupling are obvious. A small portion of code always has higher probability of correctness than a large portion of code. Second, reusability of a small portion of code is also higher than large portion of code. Third, maintenance can be easier for smaller and independent portion of code. The last benefit is for the developer, the unit testing is much smoother in a modularized application.

The MVC pattern is widely recognized as being among the most well-developed and mature design patterns in use. By using the MVC design pattern, all the classes in this project are broken into three distinct sections aptly named the Model, the View, and the Controller. Each section is decoupled into smaller modules based on the functionality. Each module, except the database connection module, can be removed without affecting the rest of the program. This also makes adding new code simpler.

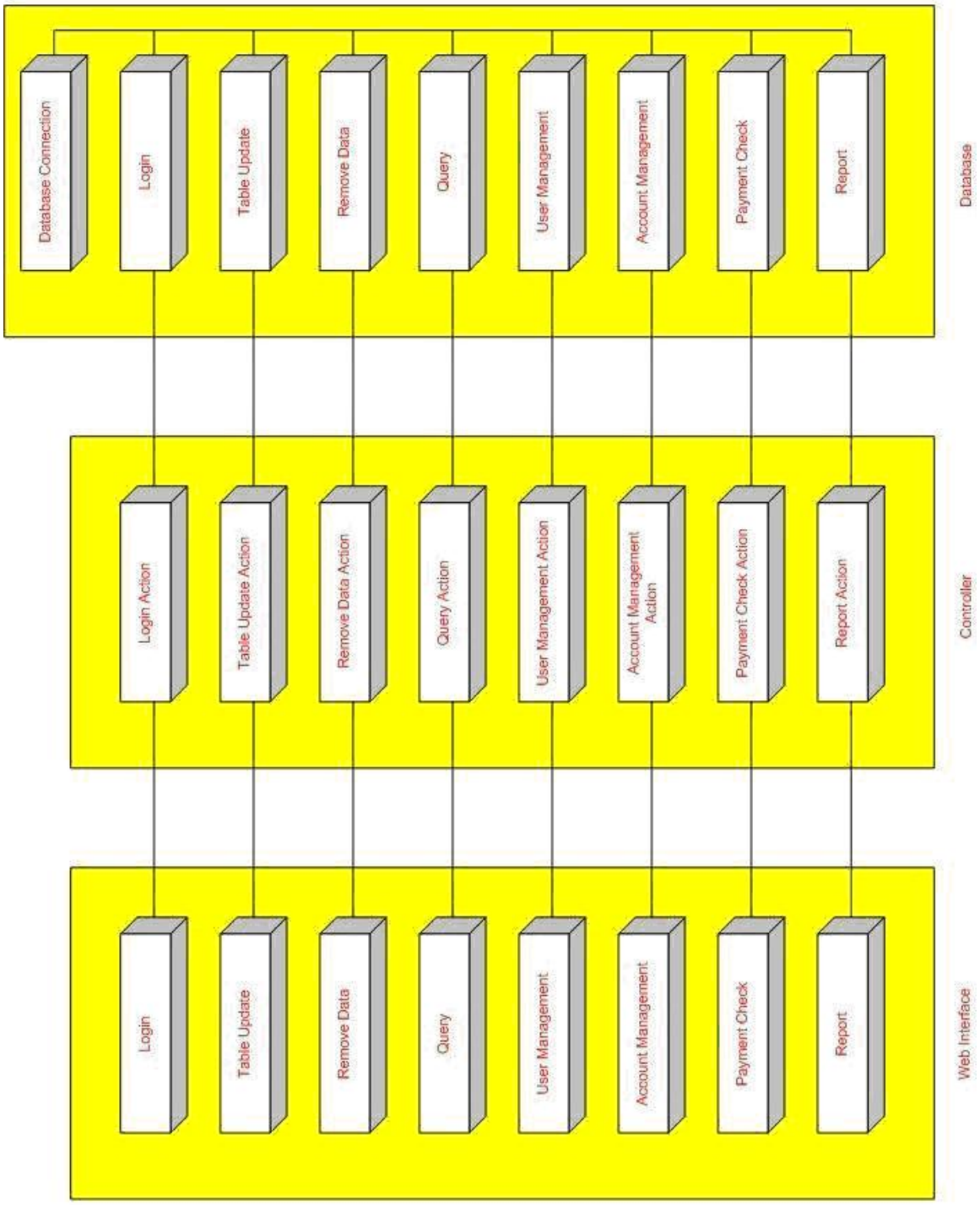

Figure 5. MVC Modules

All the modules in figure 5 are highly independent, changing, removing will not affect other modules. For example, if the report sub-module which includes Report class in the View, ReportAction class in the controller and Report class in the Model

was removed, the application will perform normally without any changes. The only task application cannot perform is the report generation and this is expected result of removing report submodule.

5.3.5 Code Reuse

Code is reused in three different ways in this application.

- Using the frameworks

  Most common parts are already implemented by the framework. Code is highly reused at least at the view layer by using tiles framework. There is actually only one JSP page developed - mainLayout.jsp. This page is used throughout the application except the login page. All the web pages are dynamically generated at the runtime by inserting defined components in the XML description file into this page.

- Inheritance

  For any classes are similar but cannot be combined into one class, defining a super class to include all the common parts and all the classes inherit from it to avoid the duplication.

- Defining more specified methods

  Any code is not strongly related to the class should be separated out into the specific class, even the code is only going to be used in one class. It can make class smaller and code also can be reused by other classes. For example, the field “parent education level” is presented as numeric in the CAS database and converted to descriptive string for better readability in the data warehouse table populating class. Although this is the only place to convert parent education level from integer number to string, a help method called getEducationLevel takes the integer as input and return the descriptive string was coded in the help class. And there are many other help methods such as getMajor, getCollege, etc. in the help classes. Those methods can be coded in the actual class, where they are called. Separating those codes into separated methods in different classes improved the

readability of the code and those methods can also be used in other programs by simply importing the help class.

5.4 Testing

Testing for this project is divided into unit test, integration test and system test.

- Unit testing

  Unit testing started at the same time coding started by using JUnit. Test cases were written for each class and each method and stubs were hard coded.

- Integration testing

  The code coverage strategy was applied in integration testing. UML state diagrams and flow chart were developed to verify code coverage testing was correctly performed. Because the project has a very simple web user interface, every function had been executed; every evaluation point was tested; every statement had been executed; all the possible path and page flow were traversed and all the possible function calls and returns were tested.

  There are many exception handlings in the application and many of them were not tested.

- System testing

  System testing was conducted to verify the requirements of the financial aid data warehouse were fully met.

The test on data warehouse correctness is much more difficult. Due to the data volume, exhaustive test is not realistic, so testing is done mainly querying aggregation results from both school database systems and the financial aid data warehouse. The results are compared and matching.

- Query total numbers for some data set, such as total number of freshmen, total number of students received certain type of financial aid, etc.
- The fiscal year 2007 Princeton Survey was already generated by ITS. This application also generated the report for the fiscal year 2007 from financial aid data warehouse and results are matching.

- Staff from both offices ran some queries against known results and checked the results of the queries to make sure the querying results are matching the known results.

5.5 Deployment

In this application, ANT script was made to auto deploy to both development and production environments. During the testing stage, ITS added the new requirement that every project has to be developed and deployed through Oracle JDeveloper. It is out of the scope of the project, but in order to successfully deploy to the server, some 60 hours devoted to this new requirement, including learning a new IDE, rearranging the file structure, configuring the Oracle Application Server, discovering the bugs in newly released Oracle Application Server and finding the solution to fix the bug. The application is deployed to both WAR file and EAR file through JDeveloper.

6. LIMITATIONS

Although the financial aid data warehouse implemented all the customer requirements and it is a good starting point to create a unified campus wide data warehouse, there are still many limitations in the application.

The first is that the updates cannot back flash to the database. It is understandable for the security reason, the original data should not be modified by any application other than its own. But no clear solution on how to correct the mistakes discovered by the application will cause problems in the future. If the CAS database mistakes have not been corrected, the application will continuously report the error and try to update already been updated data set.

The second limitation is SQL queries for the reports. Because SQL query is not object-oriented, scalability will be a problem when the reports change in the future. The current solution is to put all the queries in a file and import all the queries into the application at run time. But due to the complexity of the queries, especially aggregated queries, some complicated queries have to be broken into several small queries and the results are calculated in the code. If those queries changed, the programmer has to go to the code level to make the change.

In the application, security was considered to be highest priority, but still security loopholes are in the application, due to the mechanism of creating new user. The user name and password for the newly created user will be emailed in plain text which can be exposed to any people. One of simplest solution might be to use LDAP login. Another security issue the application trying to fix is to eliminate the plain text file stored on the local machine, but it didn’t get fixed and only gets worse, because every query result will be stored on the financial aid office local machine. It requires the financial aid office to aware this security loophole and cautiously adapt correct procedure to minimize the loophole.

7. FUTURE WORK

The application implemented all the initial requirements, but due to the knowledge limitation to the problem domain and the data warehouse design, following improvements should be added to the application in the future.

- Web User Interface for this application can be improved by a professional web designer.
- Some extremely complex queries were constructed by combining Java code with SQL. These queries should be constructed by pure SQL if possible to separate the query and code, so the query change does not require code change.
- Currently the query result set is in alphabetic order. The ideal situation is to use applet to interact with the user, giving user the ability to arrange the order, but it is a big work. It not only requires to create applet to allow the user arrange the order, but also requires modify SQL query string in the business logic to accommodate the changes.
- Error handling and Exception handling can be better than simply redirecting user to login page. Especially exceptions in the business logic should guarantee confidentiality and correctness.
- Log file can be more readable and should log more information other than just user activities and exceptions.
- Develop an exhausted test plan to test whether every record pulled from database is correct and indeed is what the application requested.
- For the code reuse, a super class or an interface for populating the data warehouse tables should be created to generalize the code. The populating table class in this application is highly specified and only works on financial data.
- Add pre-calculated fields in the fact tables.

8. CONCLUSION

Financial aid data warehouse application is a web application that provides a unified, consistent data query environment and also to provides an easy-to-use tool for financial aid office to query student financial data with minimum maintenance overhead. Populating data warehouse and querying data warehouse are implemented as two separated modules. Now financial aid office can query data only in this data warehouse without going across different databases and the financial aid office staff are satisfied with intuitive and simple web user interface.

Using the frameworks makes this application easier to maintain than without using framework. The future work list will make the application more robust and more reliable.

One of the interesting research topics arose from this project is how to reduce the technologies used in the web-application. Another interesting topic is to develop some kind of debug tool for the web application since even commercial web server such as WebLogic has no debugging assistant tool for the developer.

Many things I learned from this project. The most important is no matter what software development model used for project, the requirements analysis and design should take time to go really deep and mature before jumping into coding. Without thoroughly gathering and analyzing the requirements and detailed design, no one will be able to develop robust, reliable and error-free software.

9. BIBLIOGRAPHY

[1] C. T. Arrington, Seyd H. Rayhan, *Enterprise Java and UML* (2nd Edition), John Wiley & Sons, Inc. 2006

[2] Alex Berson, Stephen J. Smith, *Data Warehouse, Data Mining, and OLAP*, McGraw-Hill, 1997

[3] Michael R Blaha, James R Rumbaugh, *Object Oriented Modeling and Design with UML* (2nd Edition), Pearson Education, 2005

[4] Kevin, Duffey, Vikram Goyal, Ted Husted, Lance Lavandowska, Sathya Narayana Panduranga, Krishnaraj Perrumal, Joe Walnes, Richard Huss, Meeraj Moidoo Kunnumpurath, *Professional JSP Site Design Coding Core Web Applications,* Wrox Press Ltd., 2001

[5] Ramez A. Elmasri, SHamkant Navathe, *Fundamentals of Database Systems* (3nd Edition), Addison-Wesley, 2001

[6] Hans-Erik Eriksson, Magnus Penker, Brian Lyons, and David Fado, *UML 2 Toolkit,* Wiley Publishing Inc. 2003

[7] Jayson Falkner, Ben Galbraith, Romin Irani, Casey Kochmer, Sathya Narayana Panduranga, Krishnaraj Perrumal, John Timmey, Meeraj Moidoo Kunnumpurath, *Beginning JSP Web Development*, Wrox Press Ltd., 2001

[8] Marty Hall, Larry Brown, *Core Servlets and Java Server Pages Volume 1: Core Technologies* (2nd Edition), Prentice Hall, 2006

[9] Jiawei Han, Micheline Kamber, *Data Mining Concepts And Techniques*, Morgan Kaufmann Publishers, 2001

[10] Erik Hatcher, Steve Loughran, *Java Development with ANT,* Manning Publication Inc. 2003

[11] James Holmes, *The Complete Reference: Struts* (2nd Edition), McGraw-Hill Companies, 2007

[12] Mehmed Kantardzic, *Data Mining Concepts, Models, Methods, And Algorithms,* IEEE Presses, 2002

[13] Stefan Koch, *JavaScript A Programmer's Companion from Basics through DHTML, CSS and DOM,* John Wiley & Sons Ltd, 2003

[14] Phillip A. Laplante, Colin J. Neill, *The Demise of the Waterfall Model Is Imminent and Other Urban Myths*, February 2004

[15] Tom Megrino, Dori Smith, *JavaScript For The World Wide Web* (5th Edition), Peachpit Press, 2004

[16] Philip J. Pratt, *A Guide to SQL* (6th Edition), Thompson Course Technology, 2003

[17] Deborah S. Ray, Eric J. Ray, *Mastering HTML and XHTML,* SYBEX Inc., 2002

[18] Gregory D. Speegle, *JDBC: Practical Guide for Java Programmers,* Morgan Kaufmann Publishers, 2002

APPENDIX A. DATA DICTIONARY

Table Name: Student Application

	From Table	From Field	Notes
Application PK			
Year	APPLICATION DETAIL	APPLICATION YEAR	
Term	APPLICATION DETAIL	APPLICATION TERM	
Application Status	APPLICATION DETAIL	APPLICATION STATUS	
Initial Entry Status	APPLICATION DETAIL	APPLYING AS	Can rename this field to "Applying As"

Table Name: Student Demographic

	From Table	From Field	Notes
Student PK			
First Name	STUDENTMASTER	concatenate(STU-NAME-F18,STU-NAME-L12)	Student names are not split in the UWL data warehouse.
Last Name	STUDENTMASTER		
Legal Address	FIN ISIR	Filler44 94; Stu Permanent State; Filler97 101	The street address is in Filler 44 94 and consists of the first 35 characters; city is the last 16 characters in Filler 44 94; state is in Stu Permanent State; zip code is Filler 97 101
UWL ID	STUDENTMASTER	STU-ID	
SSN	STUDENTMASTER	STU-SSN	
DOB	STUDENTMASTER	STU-BIRTH-CDATE	Format will be YYYYMMDD
Gender	STUDENTMASTER	STU-SEX	
Race	STUDENTMASTER	STU-RACE	0 = Unknown; 1 = White, 2 = African American; 3 = Native American; 4 = Other Asian Pacific Islander; 6 = Southeast Asian
Ethnicity	STUDENTMASTER	STU-RACE	5 = Hispanic
Original Residency	STDACCTSUMMAR	SAS FEE STATUS	
Father ED Level			father's highest level is the middle character of this field; mother's highest level is the last character of this field; coding: 1 = Middle School/Jr. High; 2 = High School; 3 = College of Beyond; 4 = Other/Unknown; blank = No response
Mother ED Level	FIN ISIR	Filler226-228	
High School Graduate Date	STUDENTMASTER	STU-HS-GRAD-CYR	
Class Size	STUDENTMASTER	STU-HS-CLASS-SIZE	

Class Rank	STUDENTMASTER	STU-HS-RANK	
SAT	PROSPECT DETAIL	SAT VERBAL SCORE; SAT MATH SCORE	Sum of these two fields
SAT Math	PROSPECT DETAIL	SAT MATH SCORE	
SAT English	PROSPECT DETAIL	SAT VERBAL SCORE	
SAT Writing	PROSPECT DETAIL	SAT WRITING SCORE	
ACT Composite	PROSPECT DETAIL	ACT COMPOSITE SCORE	
ACT English	PROSPECT DETAIL	ACT ENGLISH SCORE	
ACT Math	PROSPECT DETAIL	ACT MATH SCORE	
ACT Writing	PROSPECT DETAIL	ACT WRITING SCORE	

Table Name: Student Financial

	From Table	From Field	Notes
Student FK			
Year	FINAIDMASTER	FIN-TERM-CFY	
Term	FINAIDMASTER	FIN-TERM-T	1= SUMMER; 4= FALL; 7=SPRING
Dependency	FINAIDMASTER	FIN-DEPENDENCY	D = DEPENDENT LIVING AT HOME; A = DEPENDENT LIVING AWAY FROM HOME; I = INDEPENDENT
Interested in Work/Study	FIN ISIR	STU WORK STUDY	1 = YES; 2= NO; BLANK = NO RESPONSE
Cost	FINAIDMASTER	FIN-COST-ATTEND	
House Size	FINAIDMASTER	FIN-HOUSE-SIZE	
Total Income	FINAIDMASTER	FIN FAMILY INCOME; FIN STUDNT INCOME	If Dependency <> Independent, then FAM; else STU
Total Contribution	FINAIDMASTER	FIN FAM CONTRIB; FIN STUDENT CONTR	If Dependency <> Independent, then FAM; else STU
Proposal Status	FINAIDMASTER	FIN-PROPL-STATUS	
Due Date	FINAIDMASTER	FIN-DUE-CDATE	
Auto0	FINAIDMASTER	FIN-AUTO-0	

Table Name: Student Academic

	From Table	From Field	Notes
Student FK			
Fiscal Year	ENROLLMENTBASE	CTERM	First four digit of CTERM
Term	ENROLLMENTBASE	CTERM	Last digit of CTERM
Major	ENROLLMENTBASE	MAJOR1	
Second Major	ENROLLMENTBASE	MAJOR2	
Third Major	ENROLLMENTBASE	MAJOR2	
Fourth Major	ENROLLMENTBASE	MAJOR2	
Fifth Major	ENROLLMENTBASE	MAJOR2	
Sixth Major	ENROLLMENTBASE	MAJOR2	
College	ENROLLMENTBASE	COLLEGE	This will only be the real college for undergrad students; see Graduate College crosswalk
Residency	STDACCTSUMMAR	SAS FEE STATUS	

Local Address	STUDENTMASTER	STU-LOCAL-ADDRESS	Again, we'll want this as separate fields
Total Semester Attended	STUDENTMASTER	STU-TRANSFX-SEM	
UWL Semester Attended	ENROLLMENTBASE	ED1-CRD-HOUR-LOAD	Computed item - add one for every unique term where ED1-CRD-HOUR-LOAD > 0
Credit Load	ENROLLMENTBASE	ED1-CRD-HOUR-LOAD	
Cumulative GPA	STUDENTMASTER	STU-HONOR-PTS/STU-CRD-ATTEMPTED	
Class Standing	ENROLLMENTBASE	ED1-YR-IN-SCH	
Student Status	ENROLLMENTBASE	ED1-NEW-CON	
Classification	ENROLLMENTBASE	Concatenate(ED1-YR-IN-SCH,ED1-COLLEGE,ED1-CLASSIFY L2)	
Official Enrolled	ENROLLMENTBASE	ED1-CRD-HOUR-LOAD	Working on getting an "Official" field If ED1-CRD-HOUR-LOAD > 0, then Y
Applied for Aid	FINAIDMASTER		if record exists, then Y

Table Name: Student Classification

Note: This Table is populated by a property file.

		Notes
Classification	Concatenate(ED1-YR-IN-SCH,ED1-COLLEGE,ED1-CLASSIFY L2)	
Title		See Graduate College crosswalk
College		See Graduate College crosswalk

Table Name: Student Major

	From Table	From Field	Notes
Major Code	ENROLLMENTBASE	MAJOR1, MAJOR2	
Description	PROGRAMMASTER	PRO CURR TITLE	
Degree Type	PROGRAMMASTER	PRO DEGREE TYPE	
Major Type	PROGRAMMASTER	MAJOR1, MAJOR2	Last digit of the major code indicates the type; 0 = 1st Major; 5 = 2nd Major; 1 = Minor; 2 = Program; 3 = Concentration; 4 = Emphasis; 7 = Associate's Degree; 6 = Certificate

Table Name: SFinancial Aid

Note: This Table is populated from a property file.

	Notes
Aid PK	
Code	See Program Codes worksheet
Name	
Amount	
Source	

Type
Needbased

Table Name: Student Term Fact Table

Student FK
Application FK
Year
Term
Major
Second Major
Third Major
Fourth Major
Fifth Major
Sixth Major
Residency
Credit Load
Accumulate GPA
Class Standing
Enrollment
Dependency
Interested in Work/Study
Cost
House Size
Total Income
Total Contribution
Auto0

Table Name: Financial Aid Fact Table

Student FK			
Year	AIDAWARDMASTER	AID-CTERM	First 4 digits of AID-CTERM
Term	AIDAWARDMASTER	AID-CTERM	Last digit of AID-CTERM
Aid FK	AIDAWARDMASTER	AID-PROGRAM-CODE	See Program Codes worksheet
Offering Amount	AIDAWARDMASTER	AID-AMT-AWARD	
Accept Amount			
Disperse Amount	AIDAWARDMASTER	AID-AMT-EXPENDED	
Lender ID			