Stephen R. Barbe. UNC Departmental Profiles Project: Designing an Online Database Retrieval and Reporting System for the UNC Department of Institutional Research and Assessment. A Master's Paper for the M.S. in I.S. degree. November, 2005. 71 pages. Advisor: Brad Hemminger

This paper describes the design, development, and testing of a web application for web reporting of institutional research data.

The project was initiated at the request of the management of the Department of Institutional Research and Assessment. IR&A is responsible for collecting, summarizing and statistically analyzing a vast amount of university data which is currently stored in a large SAS database and accessed through a SAS application. The management of IR&A felt that the current system had grown too unwieldy and sought an alternative solution for making this data accessible to their user population.

This paper describes the phases of the project, from the initial gathering of functional requirements and design of a project plan to the design and construction of a new relational database to store and the design and development of a dynamic Web-based front end for accessing and manipulating the data.

Headings:

Computers -- Electronic Data Processing

Databases – Relational Databases

Internet

Web Application

UNC DEPARTMENTAL PROFILES PROJECT: DESIGNING AN ONLINE DATABASE RETRIEVAL AND REPORTING SYSTEM FOR THE UNC DEPARTMENT OF INSTITUTIONAL RESEARCH AND ASSESSMENT

By
Stephen R. Barbe

A Master's paper submitted to the faculty
of the School of Information and Library Science
of the University of North Carolina at Chapel Hill
in partial fulfillment of the requirements
for the degree of Master of Science in
Information Science.

Chapel Hill, North Carolina

November 2005

Approved by

_____

Brad Hemminger

Table of Contents

I. Introduction

This paper will describe the design, development, testing and deployment of a web application for accessing, displaying, charting and converting institutional research data into a variety of end-user formats.

This project was initiated at the request of the management of the Department of Institutional Research and Assessment (IR&A) at UNC Chapel Hill. IR&A is responsible for collecting collating, summarizing and statistically analyzing a vast amount of data related to faculty, staff, student enrollment, credit hours and departmental budgets. This information was formerly stored in a large SAS database and accessed through a SASweb application. The management of IR&A felt that the current system had grown too large, complex and unwieldy and sought an alternative solution for making this data accessible to their user population.

The current system in place to allow users to access the data provided by IR&A stores the information in a series of SAS OLAP cubes and provides a web front end for accessing these cubes to retrieve planning and forecasting data. An OLAP (OnLine Analytical Processing) cube uses a multidimensional database model to create a database structure which combines aspects of navigational and hierarchical databases as opposed the relational database structure used by such products as Access and Oracle.[Wikipedia, 2005] The current structure of the application begins with a very high level set of report types to choose from, listed as a set of links in the center of the page. These links lead to other lists of links which break down the earlier selections into more specific subsections.

This continues, onion-like, through several more levels in order for the user to reach the report data. This structure, while organized and functional, requires users of the system to work their way from the top of the cube structure down to the level where the data they wish to find is stored using a series of drop down selection boxes which are nested within each other. The management of the IR&A department felt that this drilling down approach had become too cumbersome and difficult for the end user to make efficient use of the system. In order to address these difficulties, the Department Profiles Project was initiated.

This project involved several phases, from the initial envisioning stage to the gathering of functional requirements and design of a project plan, to the design and construction of a new relational database to store the data and the design and development of a dynamic Web-based front end for accessing and manipulating the data into various formats for presentation to the user. The initial development phase of this project was conducted in the spring 2005 semester as part of the coursework for the INLS 259 class at the School of Information and Library Science at the University of North Carolina at Chapel Hill. During this time, the project was proposed by my classmate, Kay StewartNewman at the request of her manager in the IR&A department at the university. Once proposed and accepted, the initial project vision was developed and formalized, the functional specifications for the new system were collected from the management of IR&A, several available technologies were evaluated for use in development, and two of these technologies were selected for use. After the technology decisions were made, we considered the approach we wished to take to the design given the functional specifications of the project. It was decided that I would provide E-R

diagrams for the proposed database structure and take responsibility for the design and development of the application's login, authentication and user administration systems while Kay would build the actual database and be responsible for the design and development of the data retrieval and presentation portions of the application. Proceeding with this plan, we completed development of the application prototype in May of 2005 using a small sample dataset from the SAS application and submitted it to the management of IR&A for their approval. IR&A accepted the prototype as proof of concept and engaged my services to develop the production version of the application during the summer of 2005. During this time, I was given responsibility for the continued development of all phases of the web application and was partnered with another IR&A staff member, Jiang WeiGuo, who was responsible for the migration of the full SAS datasets to the new database.

Section II of the paper will discuss the initial envisioning stage of the project, in which the problem to be solved by the project was identified, a project vision developed, and the vision of the project reconciled with the mission statement of IR&A. Section III of the paper will consist of a review of several of the available technologies which were considered for use in the design and development of this project. This section will in many ways substitute for the literature review portion which would be present in most traditional research papers. Section IV will detail the functional specifications as initially established and describe the evolution of those specifications throughout the prototype phase of application development. Section V will discuss the initial project design decision process from the standpoint of both establishing a new relational database to contain the data migrated from the SAS application and the design of a new web-based

application with which the users of the system will access this data. Section VI will discuss the evolution of the project from the initial prototype to the Beta development phase with emphasis on an expansion of the initial functional specifications, a required redesign of the user administration system, and some challenges that were encountered in the migration to a different platform than the one used for the prototype system development. Section VII will outline the testing process for the application, including developmental testing/troubleshooting, post development user testing, and additional expansions of requirements after the initial application testing phase. Section VIII will focus on some of the issues considered in the planning of application deployment and support. This section highlights the need for a User's Guide to be created for the application prior to deployment and the necessity of redesigning the interface to provide on screen documentation to guide the user. As of this time, the application is still in development, so no actual deployment has occurred. Finally, section VIIII will provide the conclusion to the paper with descriptions of where we feel the project succeeded, where we feel it failed, some lessons learned from the project, and some ideas for future work or development.

II: Envisioning

During the design and development of this project, the project team followed a standard system for planning and developing software known as the Software Development Life Cycle (SDLC) model. This model divides the process of software design into four distinct phases: Envisioning, Functional Specifications, Development, and Testing. During the envisioning phase, we made some basic conceptual decisions

about what the goals of the project were and what we hoped to accomplish with the design of this application. After considering these questions, we established a vision statement and tied that statement to the mission of the IR&A department. This vision was revised periodically during the life of the project, but the final vision statement is:

> The aim of this project is to develop a dynamic, Web-based data reporting and comparison tool featuring multi-tiered access and a customizable interface for the UNC-Chapel Hill Office of Institutional Research and Assessment.

In addition to the vision of the project, a list of business requirements and constraints were identified which impacted the functional specifications which formed the basis of many of the design decisions made during the software development phase of the project. The following requirements and constraints were identified during the project envisioning phase:

- System administrators may authorize departmental administrator access
- User and Query Data may be added by departmental administrators
- Departmental reports will be designed and added by departmental administrators
- Departmental administrators may authorize department user access
- Data may be updated or loaded by system administrator, as it becomes available to the Office of Institutional Research
- High-level administrators may authorize departmental administrator access
- Departmental administrators may authorize department user access
- System administrator may authorize access for all
- High-level administrators may authorize viewing of their departmental data for sharing with other high-level administrators

- System must be able to gather user feedback to include additional summary data or add further drilldown capability for retrieval of more detailed information
- With the exception of the one employee of the Office of Institutional Research on the project team, the Office of Institutional Research will not provide monetary compensation to team members for work on this project
- College of Arts & Science Dean's Office will have the capability of viewing summary data for the college and division.

From these requirements and constraints, the project team then constructed a list of functional specifications for the proposed application.

III: Functional Specifications

Like the vision statement, the functional specifications were revised several times during the course of the project. The first stage in establishing the functional specifications for the application consisted of constructing a simple, prioritized list of features desired in the finished application. This initial list was then reviewed by the individual project team members who each made comments and suggestions for revision of the list into its final form. The final list of functional specifications for the application was compiled with the following results:

1 Export SAS data to unified relational database management system and
Enable single query retrieval of any amount of data from any
combination of tables.

2 Provide query output in a variety of formats

a. HTML Web page: This will be the standard response to any query performed by the system.

b. CSV file: After logging into the system and choosing the departments he needs to query, the user will then select the query to perform (stored or create new) and then select their query output format from a list of choices (HTML will be default). If CSV option is selected, the query results will be converted to a list of comma separated values which the user may then save on their local PC.

c. Excel spreadsheet: Like the CSV option above except that when this option is selected, the user has the option to allow MS excel to be opened on their client machine and an excel spreadsheet created.

d. Plain text (.txt): This is the same as the CSV and Excel options, but saving the results as plain text.

e. E-mail results to x: this will email any of the above formats to a user whose email address is entered.

3 User/login authentication

a. User login will be onyen based (this is a requirement), but once the user is logged in, a user table in the db will be consulted to establish the user's access level to determine access rights and permissions within the application.

4 Design custom reports: This will allow the user to select from a list of drop down/check boxes to decide which tables they need information from and which information they need. These boxes will select the various components to construct a

query which may then be stored in a table for later use if desired. This will be done by departmental admin.

5. User administration capabilities:

   a. Add user-will allow a user to access the application's user table and add a new user who will have permissions to use the application

   b. Delete user-as above, but allowing a user to be removed from the application

   c. View user list-a query of the user table which will show user information including name, dept, access level, and other information. Perhaps several different iterations of user list query

   d. Assign permissions-will allow a user to modify the access level information for users stored in the users table

6. Manage custom reports:

   a. Delete reports-will allow a user to remove a custom report that is no longer useful or necessary

   b. Update reports-will allow a user to modify the components and function of an existing saved query. By changing the options selected and re-saving, the query will be modified and updated in the stored query management tables.

7. Annotation feature: Feedback by user, related report/saved query and date/time stamp.

Due to time constraints and the loss of a team member prior to the beginning of the development phase of the project, many of these features were heavily modified or not implemented in the prototype application. Of the formats listed in functional

specification two, only the basic HTML presentation and Excel spreadsheet format were implemented in the prototype. This was mainly due to time constraints and lack of access to some required features in the application server which was available to us at the time of the prototype development. The requirements of specification four were also heavily simplified for the prototype due to a perceived lack of need for the initially proposed scheme of customization. During development, the decision was made that the requirement of simplifying the overall operation of the application was more important than the full set of customizable options initially envisioned. In essence, it was decided that instead of offering users of the new system a large list of poorly implemented options, the application function should be streamlined to limit user options and maximize ease of use. The final option for user assigned permissions under specification five was integrated into the Administrative role as a part of the Administrator's core functionality to simplify the administrative options and to limit the number of persons capable of adding and removing users and permissions. This decision was made to simplify the process of auditing and accountability for changes made to the user permissions function of the application. Specifications six and seven were dropped completely from the scope of the prototype design. The decision to drop specification six was made for much the same reason as the decision to streamline the requirements of specification four. Making all of the reports dynamic queries built on demand at runtime helped to maximize the flexibility of the data that a user can retrieve while minimizing the amount of options that they need to wade through in order to generate the reports they desire. Specification seven was dropped solely due to lack of time available at the end of the prototype phase, and has not since been implemented in any meaningful way.

IV. Technology Review

Given our list of functional specifications, we next needed to decide how best to develop our application to meet the expressed requirements. Given that neither of the project team members had had extensive experience with any type of programming or application development, we decided to evaluate a selection of different technologies for use in or application development framework. During this process, three database solutions and three application server technologies were investigated and one of each was finally chosen for use in the development of this project.

The three database solutions investigated were Microsoft Access XP, MySQL and Oracle 9i. Each of these systems provides a SQL-compliant relational database which allows access by multiple networked users.

Microsoft Access was the first database system considered for use in the development of this application. Access had much to offer in the areas of ease of use and team familiarity, but fared less well in the areas of scalability and performance. Both members of the project team had used Access in the past for the creation of small-scale databases for various purposes, which made us relatively comfortable with the idea of using Access for the database development portion of this project. Access also provides an easy to use graphical interface for performing all of the tasks involved in creating, populating and querying the database. This GUI was a major point in Access' favor due to the fact that neither of the other systems evaluated provides any GUI functionality at all by default. Included in this interface is a "Query Builder" function which allows the construction of SQL queries by the use of a wizard which walks you through the process

of selecting the tables, fields and conditions with which to construct your query. This function was found to be useful in the process of designing queries due to the ability to examine the SQL statements constructed by the wizard. Another useful feature of the Access interface is the ability to create a visual diagram of the relationships between the tables in the database.

Despite Access' strong visual interface, it suffered badly in the areas of performance and scalability. Access is not designed to be a true client/server database system, instead using a file-system database model. This model requires each user of the database to access the same physical copy of the database file in order to process queries or changes. [Chigrik, 2003] This limitation disqualifies Access from serious consideration in situations where the database will be accessed by more than a handful of people (perhaps 5-15). The database and table size limitations present in Access were also unsuitable for the size of the system that we were required to build.

The second product evaluated for use in this project was MySQL. MySQL is an open source SQL compliant client/server database which is available as a free download from www.mysql.com. MySQL has many advantages, but the version we tested had one crippling disadvantage that almost immediately pushed it out of consideration.

MySQL is a highly scalable system which has grown increasingly popular due to its high retrieval speeds, free availability and easy integration with such as other web technologies as the Apache web server and the PHP application server. These three products, which are all open source and freely available, provide a very flexible, powerful and cost effective web application framework. MySQL has the capacity to handle extremely large databases, provides very fast retrieval and query processing and supports

a wide variety of operating systems, web servers and application servers. [MySQL, 2005] The fact that MySQL is open source and free was also a very attractive feature.

Unfortunately, the version of MySQL that was investigated for use in this project does not support foreign key constraints, views or auditing, which were required for the database portion of this project. This lack of support, like Access' scalability and performance problems, meant that MySQL could not be used for the development of this application.

The third database system investigated for this project was Oracle 9i. Oracle provides all of the functionality, capacity and scalability required by the application to be developed for this project, but suffers due to the lack of GUI interface. Oracle's client/server architecture is highly scalable, providing support for multiple thousands of users and file sizes up to 64GB, which is more than sufficient for the requirements of this project. [Foch, 2002] Oracle also supports all of the functions, constraints and data types required to implement the database portion of the application.

Oracle's biggest drawback is the lack of a convenient user interface. The SQL Plus development tool that ships with Oracle is a command line interface with very few advanced features. It does not support multiple development windows, has almost no advanced editing features, and does not even support cutting and pasting of text particularly well. The command set for the interface is not very intuitive and is time-consuming and difficult to work with. In order to alleviate this problem, third party Oracle development tools were investigated, several of which proved to be significant improvements over the Oracle SQL Plus environment. One of the team members decided to use a tool called SQL worksheet, while another used a trial version of a tool called

TOAD. The availability of these alternate development tools made Oracle a much more attractive option by greatly improving ease of use for Oracle's database system.

Another factor in Oracle's favor was its ready availability and the high level of support available for it in the university's computing environment. We were provided with a test development server for classroom assignments and were also able to secure space on a pre-existing production Oracle server provided and supported by the university for the eventual deployment of the finished application. This was an extremely important consideration in the decision of which database system to use, due to the fact that Oracle is a very expensive piece of software which the project team could not have afforded to purchase had it not been already available under license from the university.

Given the strengths and weaknesses of each of the systems evaluated, Oracle was the clear choice for use in this project. Oracle provided the best combination of performance, scalability, features, support and availability for the purposes of this project. The use of third party Oracle development tools helped to bridge the usability gap between Oracle and Access, while the free availability of Oracle servers in the university's computing system nullified the cost advantages of MySQL and Access (MySQL is free and Access comes bundled with the Microsoft Office suite).

After choosing Oracle as the database system for use in this project, we next evaluated three dynamic web application server technologies for developing the web interface. Due to their ready availability, popularity and usage in the Web Databases course, the three technologies investigated were Microsoft's Active Server Page technology, PHP Hypertext Preprocessor, and ColdFusion Markup Language (CFML).

Active Server Pages (ASP) is a dynamic technology for web applications offered by Microsoft through their Internet Information Server (IIS) product. Active Server Pages can use a variety of programming and scripting languages to access data and provide dynamic access to it via the WWW. ASP embeds dynamic script into html pages using VB Script as its default development language, but provides support for a multitude of other languages, including java, javascript, perl, python and many others. [Microsoft, 2005] ASP proved to be relatively easy to learn and use, requiring only a basic understanding of VB script in order to begin developing dynamic content. The flexibility of being able to choose alternate scripting languages for use in ASP development is also a very attractive feature which unfortunately none of the project team members (who had very little previous programming/scripting experience) were able to exploit effectively.

Despite its ease of use and ready availability, ASP does have some serious drawbacks. First, ASP is tied to the use of Microsoft's IIS which eliminates the option of choosing a different web server technology. This is problematic due to the fact that IIS is not a free product, and can in fact be quite expensive depending on the number of servers and licenses required to support and access the web applications hosted on it. Cost did not play a significant role in our decision in this case due to the fact that we had access to a departmental web server that was already installed and running IIS, however the lack of availability of an IIS production server to host the finished application did weigh heavily on our choice. In addition to this, ASP and IIS have had an extremely spotty history of bugs, glitches and security holes which make them a somewhat questionable choice for developing applications which will provide access to confidential information.

The second technology examined for use was the PHP Hypertext Preprocessor (PHP). PHP is an open source scripting language which is freely available for download from www.php.net.

PHP uses special starting and ending tags to embed dynamic scripts directly in the body of html documents and provides support for dynamic web content on a variety of platforms including linux, UNIX Windows and MacOS. [Hojtsy, 2005] PHP is one of the most popular dynamic web content technologies in existence due to its open source nature, ease of integration with other popular technologies such as Apache and MySQL, and free availability. These qualities make PHP an exceptionally good choice as a dynamic web application framework for any individual or organization which needs to develop such an application with little or no monetary investment in their web technologies. PHP integrates particularly well with MySQL, as it includes built in functions for the query and manipulation of data within MySQL databases.

Despite all of these advantages, PHP was not very attractive for use in this project due to the relatively high learning curve required to begin effective development using it. As has been stated before, neither of the project team members had much programming experience prior to beginning this project, and of the three technologies surveyed for the web development, PHP was by far the most difficult to learn. The very flexibility and power which makes PHP so popular lends it a much greater complexity than that found in either ASP (VB Script) or ColdFusion. Given the time constraints on the development phase of this project, it was felt that there was simply not enough time for the development team to become sufficiently comfortable and proficient with PHP to select it as the web development language for this application. Additionally, the fact that Oracle

was selected for the database portion of the application instead of MySQL made PHP somewhat less attractive given that we could not take advantage of PHP's built in MySQL functions.

Finally, we investigated Macromedia's ColdFusion Markup Language (CFML). CFML requires an application server to process the dynamic pages created with it. The two application servers currently available for processing CFML are Macromedia's ColdFusion MX and New Atlanta's Blue Dragon server. Blue Dragon is a freely available product which provides most of the functionality of Macromedia's ColdFusion server and allows the use of the ColdFusion tag scripting language. As with IIS, a Blue Dragon server was in place and available to us for use in the development of this project.

ColdFusion is a tag-based, customizable and extensible web scripting language which supports dynamic retrieval and presentation of data through a web application. [Macromedia, 2005] Blue Dragon supports a number or operating systems, including linux, Windows, MacOs and Solaris, as well as a number of web server products including IIS, Netscape Enterprise server, and Apache. This puts it somewhere in the middle of the spectrum between ASP and PHP in terms of flexibility in choosing the platform and server architecture for use in the development framework. ColdFusion is a tag-based scripting language which is extremely similar to html in many ways. By use of special cfml (ColdFusion Markup Language) tags and special characters (notably the # character) you can insert code into the body of html pages to retrieve and display data dynamically. ColdFusion is designed to be highly modular, allowing you to build small ColdFusion pages which perform common tasks and reuse them throughout an application by providing them with different dynamic data to be used in page processing.

ColdFusion is also customizable and extensible through the use of custom tags and ColdFusion Modules. The custom tag feature allows the creation of user-defined cfml tags consisting of blocks of cfml code to perform specific functions. These blocks of code can then be named by the user and called from the body of a document just like any other cfml tag. This requires some special configuration on the server, but once enabled, allows custom tags to be stored in a separate folder on the server and used in the application. ColdFusion is also very easy to learn given its remarkable similarity to html, but does have some very confusing syntax in places, especially in the use of single and double quotation marks which are often apparently interchangeable, but sometime causes errors when the wrong ones are used.

Blue Dragon's major weakness is the lack of support for many of the advanced features which the Macromedia ColdFusion server supports. Blue Dragon does not provide support for such advanced functions as generating output in various formats (notably Microsoft Excel and Adobe PDF) and creating charts based on query data, both of which were desirable functions for the development of this application. This was not as much of a problem as it could have been, however, due to the fact that custom tag functionality allowed the use of custom-designed tags to perform some of these functions. Macromedia's server supports a great deal more functionality than the Blue Dragon server, but carries a correspondingly high price tag. Once again we were fortunate to have access to the Blue Dragon server for the prototype design of this application and to a Macromedia server already in place within the university for the hosting of the production application when it was completed.

Weighing the advantages and disadvantages of each of the technologies, it was decided that ColdFusion provided the best combination of features, availability and ease of use for this project. ColdFusion supports more choices of operating systems and web servers than ASP and was much easier to learn and develop in than PHP. The customizability and extensibility built into ColdFusion also provided us with relatively easy access to functions and custom tags to address most of the features lacking in the Blue Dragon server. The choice of Oracle as the database environment also swayed the decision significantly since it was then impossible for us to take advantage of the built-in integration features between PHP and MySQL. Finally, the fact that we had free access to all of these technologies in the university environment allowed us to choose the technologies we thought best suited to the project and the team members without the need to consider the monetary cost of the software for the development.

V. Prototype Design and Development

Once the technologies had been evaluated and the development environment chosen, the design and development phase of the project began. The list of functional specifications generated during the envisioning and planning stages of the project immediately suggested that splitting the design and development of the system into two parts would be an effective way to manage the required tasks and avoid confusion on the part of the team members. In order to implement this plan, we first identified the separate tasks that needed to be performed and then assigned them to individual team members for completion.

The first set of tasks defined for the development phase of the project were to define the characteristics of the data that needed to be stored, create an E-R diagram for the database, and design and create the tables used for storing the data.

Because she was the most familiar with the system currently in use by IR&A, Kay StewartNewman was assigned the task of defining the data to be stored. Having access to the current SAS database, she obtained copies of the schema in use within SAS. She also constructed sample datasets based on the data currently present in the SAS system. Using this information, she provided a detailed explanation of the data to be stored and the relationships between the various tables proposed for construction.

This information allowed me to complete the second task, creating our E-R diagram (See Appendix I). The E-R diagram turned out to be somewhat unusual compared to other E-R diagrams I have seen given the somewhat unusual characteristics of the data and query requirements. Because the data is statistically pre-summarized before being entered into the database and the data is always displayed only according to the academic year, the department summary data table seems to have its rows and columns reversed. This structure is extremely non-intuitive, and was the subject of much discussion during the diagramming phase of the design. Once it was clear how the actual generation of the reports needed to function, the design proved to be very sensible. This was only possible due to the fact that the system was not intended to provide access to raw data, but only to data that has already been statistically analyzed and formatted to some extent for presentation.

After the E-R diagram was complete, the database tables were logically divided and assigned to team members for creation, testing and sample population. I was

assigned the tables which constituted the user authentication portion of the system, while Kay was assigned the tables containing the necessary data for the reporting system. These tables were created using text files containing standard SQL DDL table create and insert statements to create the tables and load them with datasets for testing. Test queries were then constructed and run against the database to verify that table relationships and constraints were functioning properly, and that the proper data could be retrieved for the database in the proper ways. The fields for the Oracle database were also designed to use the same or equivalent data types as the current SAS database, allowing for simplified export of data from SAS and import of this data into the Oracle system in accordance with functional specification 1 (see section III).

Once the tables were completed and populated with test data sets, Kay began working on the design of a modular query using variable values which would later be populated using select boxes generated by the application's web front end. This query basically functions as the application's reporting engine, retrieving the requested data based on the parameters entered by the system user. The system also required the creation of some smaller modular queries for use in generating the lists which populate the select and drop down boxes which feed the main reporting query. In all cases, the queries were designed to provide high re-usability throughout the application while requiring little or no modification when new data is added to the database. This design principle also allows the user to customize the reports they generate from session to session without having to learn how to use SQL.

With the database design and creation complete, the next phase of the project was the design and development of the user interface. When considering the design

architecture of the user interface, two primary goals were identified: the application was required to provide a user authentication and administration function, and to simplify and allow customization of the report generation process for IR&A data. This definition was again used to divide the design and development tasks between the project team members. I was assigned responsibility for the user authentication and administration subsystem, while Kay was given the data retrieval and presentation subsystem. These subsystems were then designed in accordance with the functional specifications detailed in section III.

In designing the authentication and administration subsystem, functional specifications 3 and 5 were my primary concern. In order to meet the login requirements, we contacted the UNC onyen administrator and requested information on how to integrate the use of the onyen authentication system into a web application. The administrator, Todd Lewis, provided us with a link to a Perl script provided by the onyen staff for use in web page authentication. This script was modified by both myself and Kay and integrated into the application framework so that whenever any page in the application is loaded, the application checks for the presence of an onyen authentication cookie. If a cookie is found, the requested page is loaded, while if a cookie is not found, the application calls the Perl script and requires the user to login using their onyen and password. Successful authentication then sets the proper cookie for the user and serves up the application main menu. This authentication method, along with the implementation of session variables in the ColdFusion environment also ensures that the user will be expelled from the system after a certain period of inactivity, thus helping to limit unauthorized access by ensuring that the application does not allow a user to remain

permanently logged in. ColdFusion session variables allow the application to store dynamically generated data and preserve the state of that data from page to page, eliminating the need to regenerate form or query data multiple times. This functionality figures prominently in every phase of the application design.

The user administration portion of the application was designed using a very simple two table structure to store user information and assign system permissions. The first table in this structure, the Users table, consists of three required fields: onyen, first name, and last name. The users table is used to verify that the onyen used during the authentication process is that of a valid system user, and to return the name of the user for display within the application.

The second table, Deptusers, originally consisted of three fields as well: onyen, department number, and access level. The original access level concept for the prototype included three access levels: System Administrator, Department Administrator and User. In this model, the System Administrator was allowed access to all data stored in the system and has permissions to add, modify or delete users in the system. The Departmental Administrator was given limited access to the user administration functions of the system, allowing those with this role to add and remove users only for the departments to which they had access. Departmental Administrators were also granted the ability to create reports for these departments. The User role granted no access at all to the user administration system, and allowed reporting only for those departments to which the user was granted access. In order to facilitate this administrative model, the deptusers table included records only for users whose role was not System Administrator. Departmental Administrators and Users had a record in this table for each department to

which they had access, using the combination of their onyen and the department number as a compound primary key. This allows the application to retrieve the full list of accessible departments during the authentication process and to store this list in a session variable for use in constructing the drop down boxes presented to the user through the web interface. This model also allowed for Departmental Administrators and Users to have multiple roles within the application, i.e. User in one department and Departmental Administrator in another. A list of all current users of the system and their authorized departments and access levels was then easily generated by querying these two tables and displaying the results of this query in a select box which could then be used to select users from that list for modification. This model worked well for the prototype development, and fulfilled all of the administration requirements listed in functional specification 5 (see section III).

In the design of the reporting subsystem, Kay needed to consider how best to fulfill functional specifications 2, 4 and 6 (see section III). In the initial envisioning process for the application, it was thought that the system should allow for users to save the individual queries which they dynamically created during their use of the system. The reasoning behind this concept was that in any system of this nature, there would be tasks which were performed exactly the same way each time and repeated periodically over long periods of time, but that only the users of the system would really know what these tasks were. Initially this lead to the belief that a system for generating and storing "canned queries" would be beneficial to the day to day use of this system by circumventing the query construction phase of reporting and allowing the user to simply select a complete query from a list of pre-defined options. In the actual prototype, Kay

managed to simplify the report construction dialog to such an extent that this functionality proved to be superfluous in most ways, and this specification (6) was dropped from the development goals of the application. By designing a simple two step query construction dialog in which the user first selected the department from which to generate the report, and then selected the years and data categories desired, it was found that any possible query could be regenerated with little enough effort on the part of the user that the idea of “canning” commonly used queries made little sense and would only complicate the process of using the system. This design also flattened the reporting system structure immensely and eliminated the SAS system’s need to drill down from the very top level of the university hierarchy (the college level) through multiple administrative levels in order to reach the data from a single department.

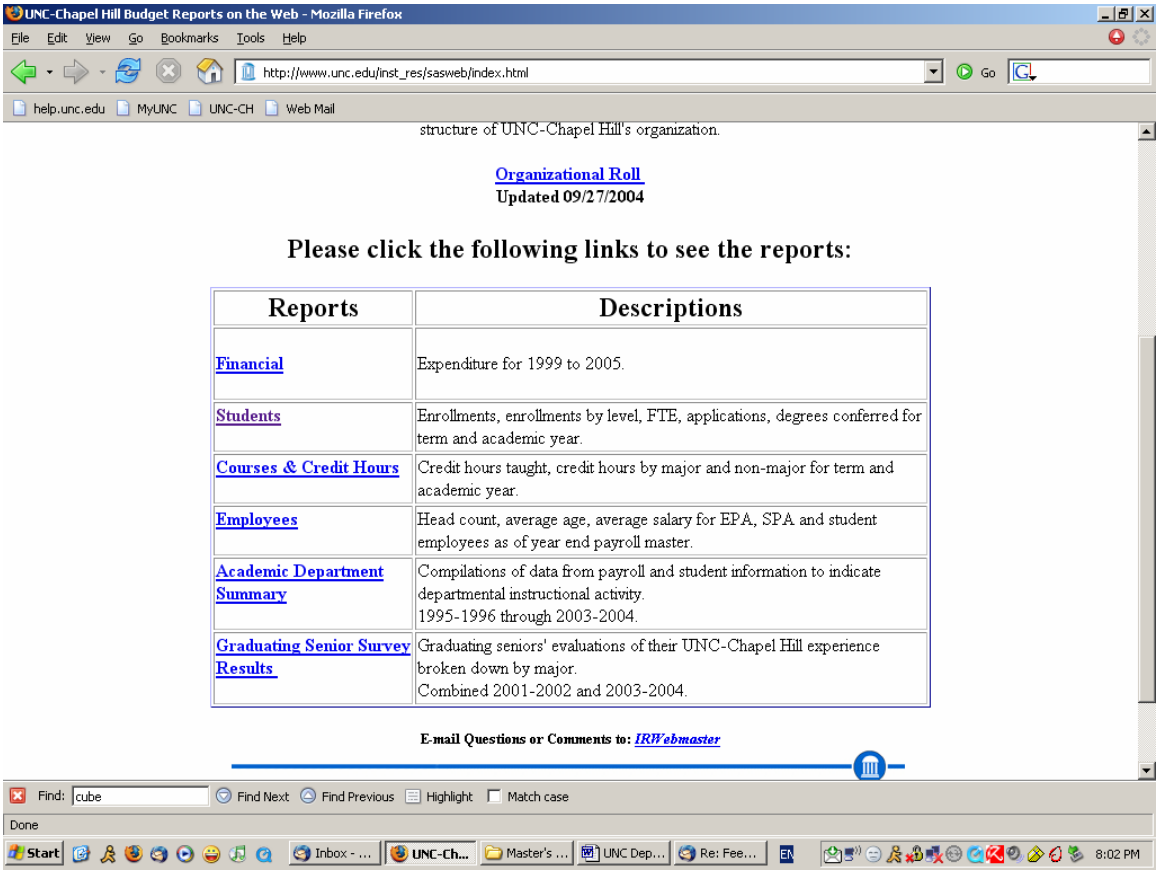


Figure 1 - SAS system main menu

Figure 1 shows the Main menu page from the current SAS application. The categories presented by the links in the left column of the table represent collections of metrics dealing with various aspects of the university's departmental structure. The links included do present a somewhat more sophisticated list of data categories than are available in the application developed for this project, but it was decided during our application design process to focus on organizing the reporting structure along the lines of the individual department instead of the type of data desired. This decision was made due to the fact that many of the users of the system work for only one department, and often wish to report on all data for that department instead of single categories or measures. Selecting "Students" on this menu produces the screen seen in figure 2.

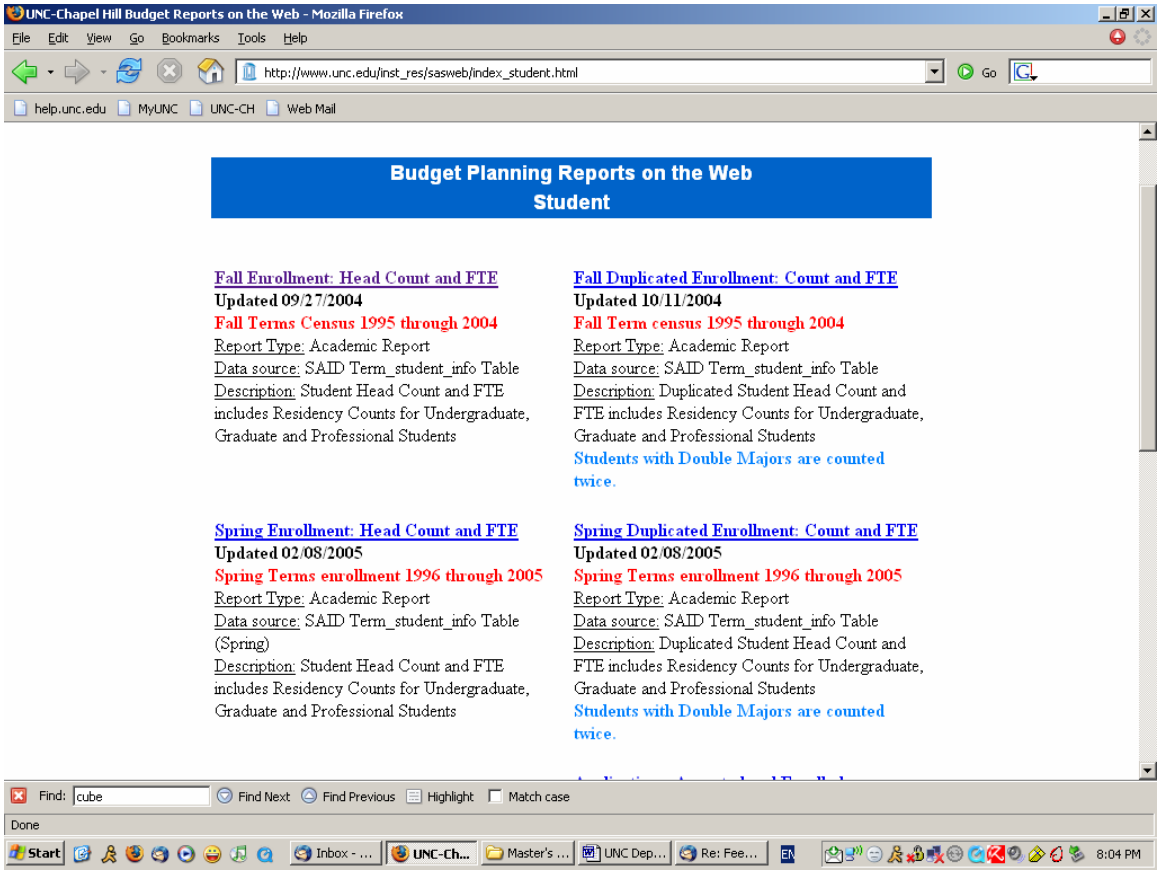


Figure 2 - screen presented after selecting "students" in figure 1

This screen presents another list of data categories broken down by term. The term separation was deemed unnecessary in the goals of our application development, and thus this screen is superfluous in terms of our design. Selecting "Fall Enrollment Head Count and FTE" produces figure 3.

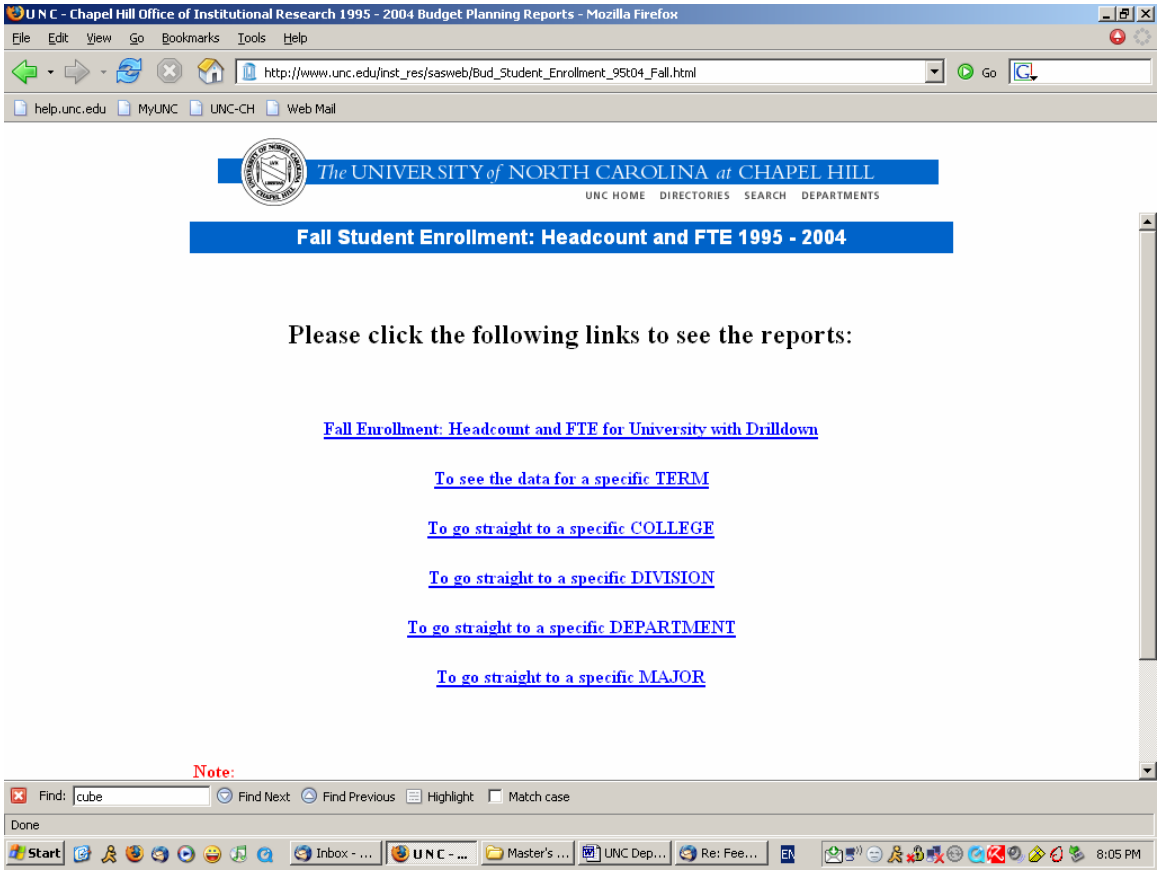
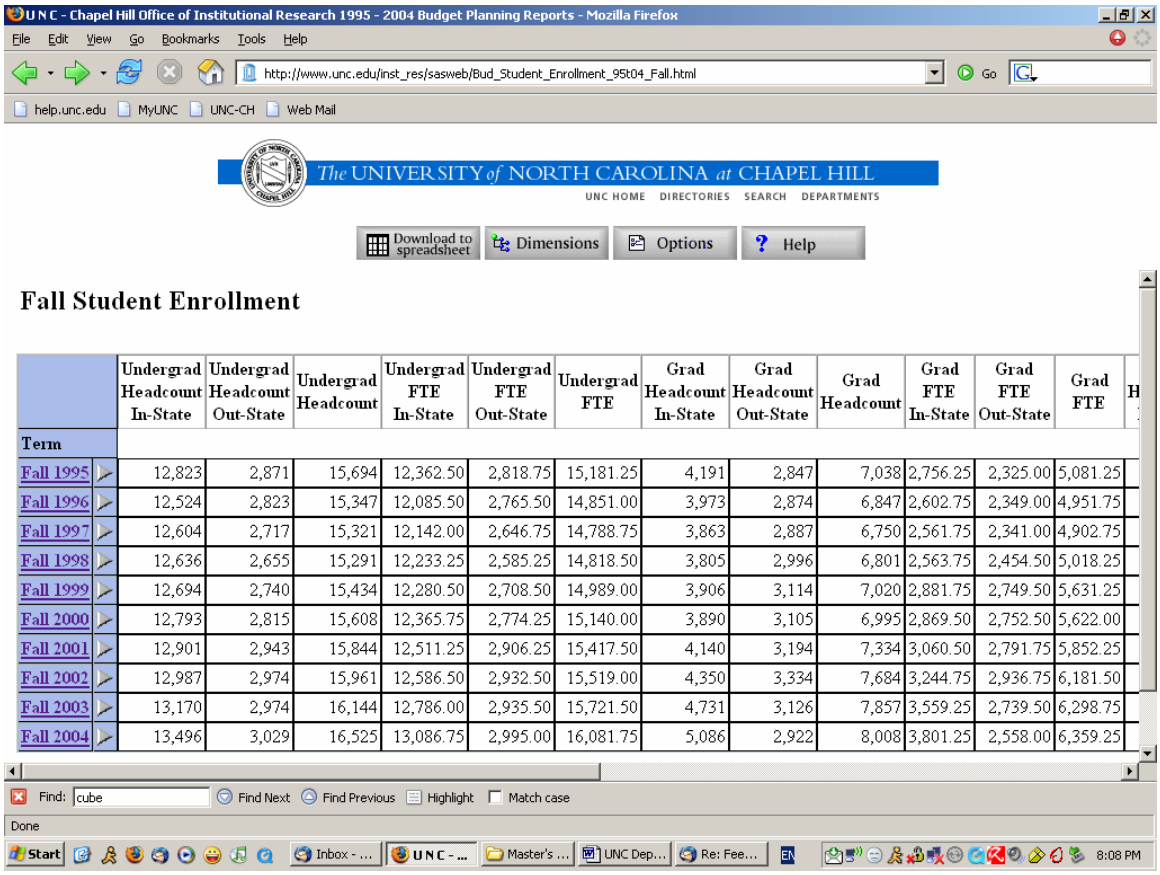


Figure 3 - Screen presented after selecting "Fall Enrollment: Head Count and FTE" from figure 2

This page provides links for accessing the selected information in multiple ways. The top link provides high level access to the database and allows the user to drill down to the specific department or major they wish to view data for. The links that follow provide the ability to pick a specific term, college, etc. and enter the database at that level through the use of drop down boxes. It was felt during the initial design and development cycle that multiple categories and entry points were causing confusion on the part of users and that this confusion outweighed the flexibility offered by the interface. This opinion has since been reviewed by the management of IR&A, and some of the rollup data

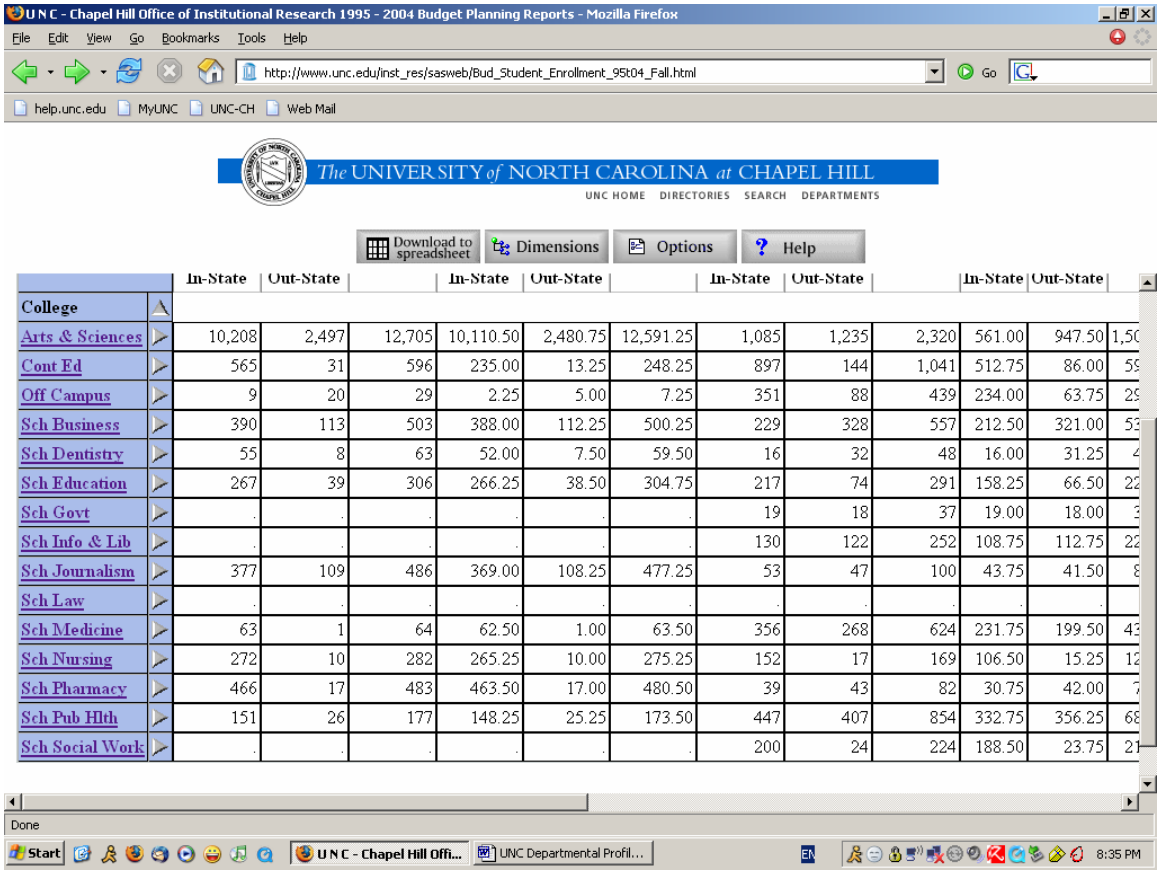
calculation provided in the current SAS application is being investigated for inclusion in the project application. Selecting the top link on this page results in the screen seen in figure 4.



Term	Undergrad Headcount In-State	Undergrad Headcount Out-State	Undergrad Headcount	Undergrad FTE In-State	Undergrad FTE Out-State	Undergrad FTE	Grad Headcount In-State	Grad Headcount Out-State	Grad Headcount	Grad FTE In-State	Grad FTE Out-State	Grad FTE
Fall 1995	12,823	2,871	15,694	12,362.50	2,818.75	15,181.25	4,191	2,847	7,038	2,756.25	2,325.00	5,081.25
Fall 1996	12,524	2,823	15,347	12,085.50	2,765.50	14,851.00	3,973	2,874	6,847	2,602.75	2,349.00	4,951.75
Fall 1997	12,604	2,717	15,321	12,142.00	2,646.75	14,788.75	3,863	2,887	6,750	2,561.75	2,341.00	4,902.75
Fall 1998	12,636	2,655	15,291	12,233.25	2,585.25	14,818.50	3,805	2,996	6,801	2,563.75	2,454.50	5,018.25
Fall 1999	12,694	2,740	15,434	12,280.50	2,708.50	14,989.00	3,906	3,114	7,020	2,881.75	2,749.50	5,631.25
Fall 2000	12,793	2,815	15,608	12,365.75	2,774.25	15,140.00	3,890	3,105	6,995	2,869.50	2,752.50	5,622.00
Fall 2001	12,901	2,943	15,844	12,511.25	2,906.25	15,417.50	4,140	3,194	7,334	3,060.50	2,791.75	5,852.25
Fall 2002	12,987	2,974	15,961	12,586.50	2,932.50	15,519.00	4,350	3,334	7,684	3,244.75	2,936.75	6,181.50
Fall 2003	13,170	2,974	16,144	12,786.00	2,935.50	15,721.50	4,731	3,126	7,857	3,559.25	2,739.50	6,298.75
Fall 2004	13,496	3,029	16,525	13,086.75	2,995.00	16,081.75	5,086	2,922	8,008	3,801.25	2,558.00	6,359.25

Figure 4 - Screen presented after selecting " Fall Enrollment: Headcount and FTE for University with Drilldown" in figure 3

As we can see, this is the first screen in the application to display actual data to the user. This table summarizes the metrics along the top for the entire university by the terms along the left side. This level of summarization is unavailable in the application developed by the project team, but once again is being re-evaluated for inclusion by IR&A. Selecting Fall 1995 from this table results in the screen shown in figure 4.

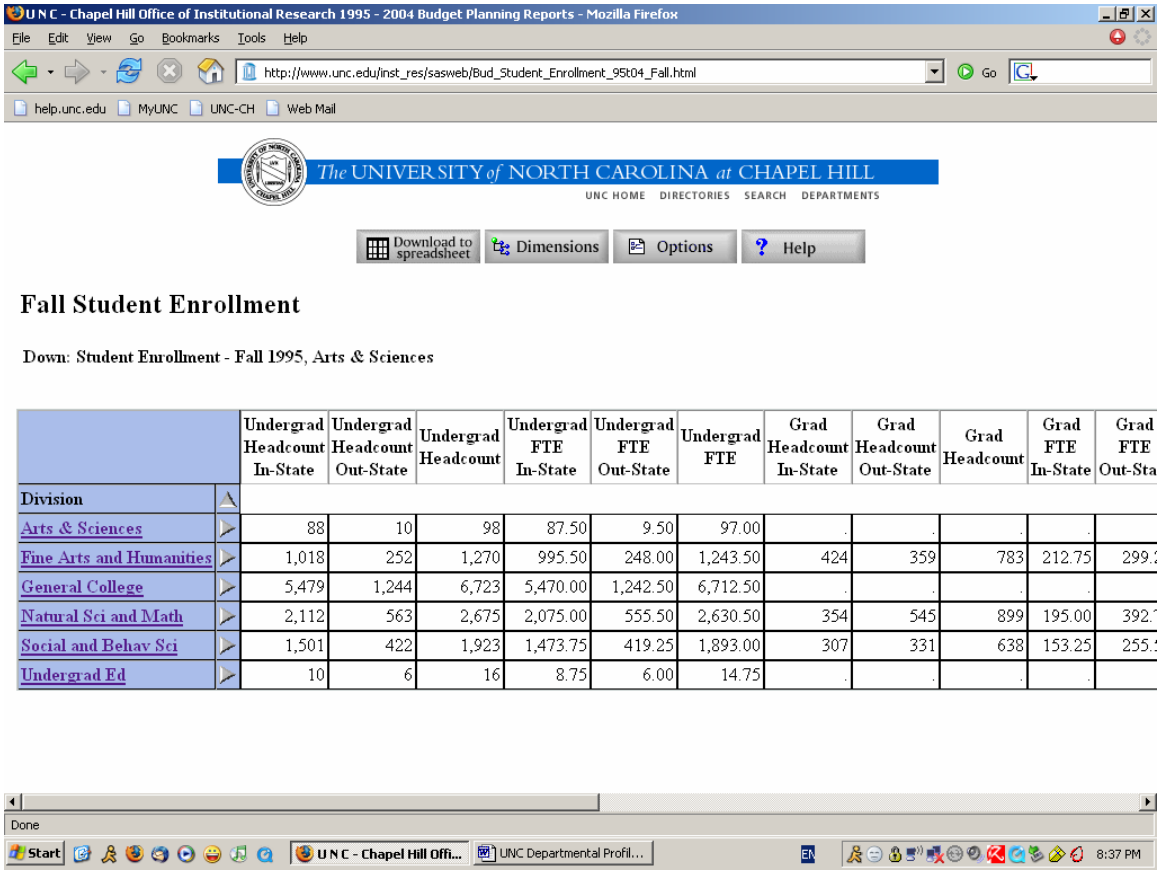


College	In-State	Out-State		In-State	Out-State		In-State	Out-State		In-State	Out-State	
Arts & Sciences	10,208	2,497	12,705	10,110.50	2,480.75	12,591.25	1,085	1,235	2,320	561.00	947.50	1,50
Cont Ed	565	31	596	235.00	13.25	248.25	897	144	1,041	512.75	86.00	59
Off Campus	9	20	29	2.25	5.00	7.25	351	88	439	234.00	63.75	29
Sch Business	390	113	503	388.00	112.25	500.25	229	328	557	212.50	321.00	53
Sch Dentistry	55	8	63	52.00	7.50	59.50	16	32	48	16.00	31.25	4
Sch Education	267	39	306	266.25	38.50	304.75	217	74	291	158.25	66.50	22
Sch Govt	.	.	.	.	.	.	19	18	37	19.00	18.00	3
Sch Info & Lib	.	.	.	.	.	.	130	122	252	108.75	112.75	22
Sch Journalism	377	109	486	369.00	108.25	477.25	53	47	100	43.75	41.50	8
Sch Law	.	.	.	.	.	.	.	.	.	.	.	
Sch Medicine	63	1	64	62.50	1.00	63.50	356	268	624	231.75	199.50	43
Sch Nursing	272	10	282	265.25	10.00	275.25	152	17	169	106.50	15.25	12
Sch Pharmacy	466	17	483	463.50	17.00	480.50	39	43	82	30.75	42.00	7
Sch Pub Hlth	151	26	177	148.25	25.25	173.50	447	407	854	332.75	356.25	68
Sch Social Work	.	.	.	.	.	.	200	24	224	188.50	23.75	21

Figure 5 - Screen presented after selecting "1995" in figure 4

This screen provides a breakdown of the data shown in the last screen by college.

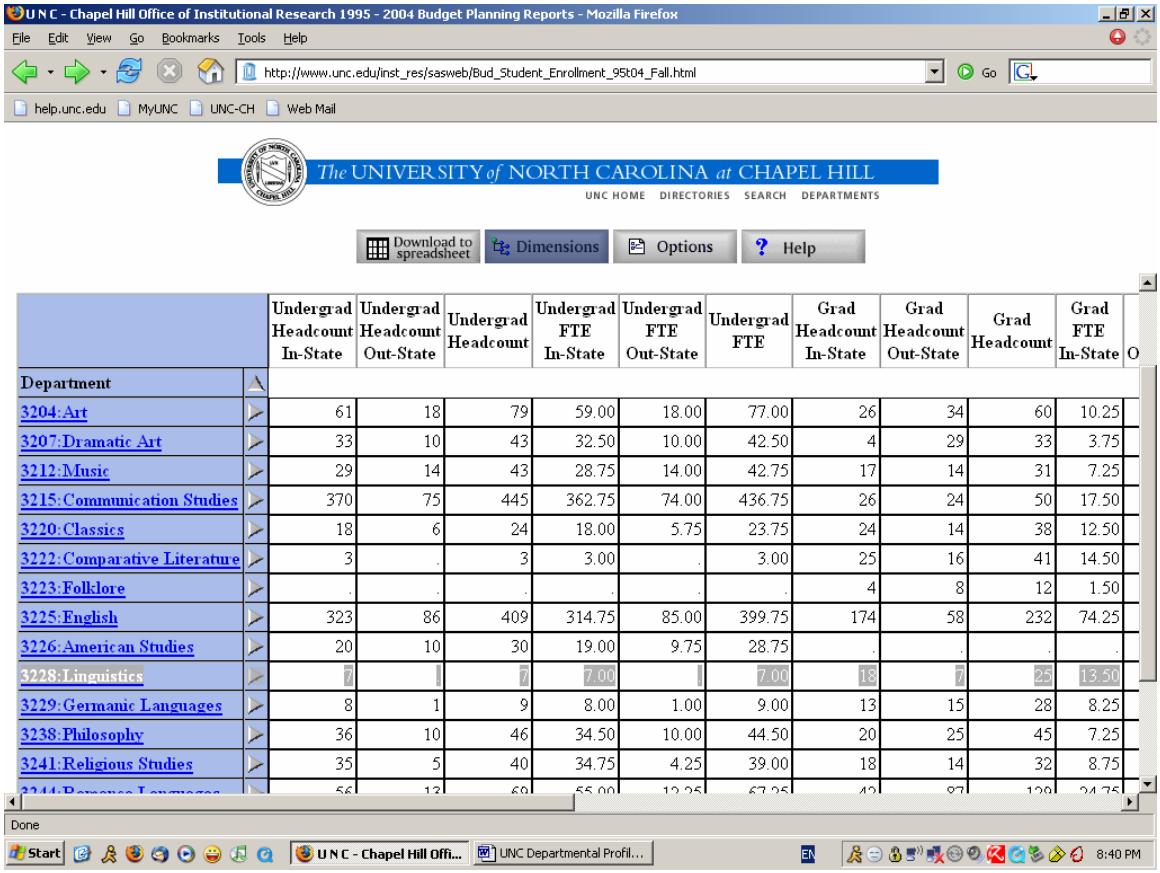
Selecting “Arts & Sciences” provides the view shown in figure 6.



Division	Undergrad Headcount In-State	Undergrad Headcount Out-State	Undergrad Headcount	Undergrad FTE In-State	Undergrad FTE Out-State	Undergrad FTE	Grad Headcount In-State	Grad Headcount Out-State	Grad Headcount	Grad FTE In-State	Grad FTE Out-Sta
Arts & Sciences	88	10	98	87.50	9.50	97.00	.	.	.	.	
Fine Arts and Humanities	1,018	252	1,270	995.50	248.00	1,243.50	424	359	783	212.75	299.
General College	5,479	1,244	6,723	5,470.00	1,242.50	6,712.50	.	.	.	.	
Natural Sci and Math	2,112	563	2,675	2,075.00	555.50	2,630.50	354	545	899	195.00	392.
Social and Behav Sci	1,501	422	1,923	1,473.75	419.25	1,893.00	307	331	638	153.25	255.
Undergrad Ed	10	6	16	8.75	6.00	14.75	.	.	.	.	

Figure 6 - Screen presented after selecting "Arts & Sciences" in figure 5

This screen presents a further breakdown of the data by division within the college of Arts & Sciences. Selecting “Fine Arts and Humanities” from this screen takes us to figure 7.



Department	Undergrad Headcount In-State	Undergrad Headcount Out-State	Undergrad Headcount	Undergrad FTE In-State	Undergrad FTE Out-State	Undergrad FTE	Grad Headcount In-State	Grad Headcount Out-State	Grad Headcount	Grad FTE In-State
3204:Art	61	18	79	59.00	18.00	77.00	26	34	60	10.25
3207:Dramatic Art	33	10	43	32.50	10.00	42.50	4	29	33	3.75
3212:Music	29	14	43	28.75	14.00	42.75	17	14	31	7.25
3215:Communication Studies	370	75	445	362.75	74.00	436.75	26	24	50	17.50
3220:Classics	18	6	24	18.00	5.75	23.75	24	14	38	12.50
3222:Comparative Literature	3	.	3	3.00	.	3.00	25	16	41	14.50
3223:Folklore	.	.	.	.	.	.	4	8	12	1.50
3225:English	323	86	409	314.75	85.00	399.75	174	58	232	74.25
3226:American Studies	20	10	30	19.00	9.75	28.75	.	.	.	.
3228:Linguistics	7	1	7	7.00	1	7.00	18	7	25	13.50
3229:Germanic Languages	8	1	9	8.00	1.00	9.00	13	15	28	8.25
3238:Philosophy	36	10	46	34.50	10.00	44.50	20	25	45	7.25
3241:Religious Studies	35	5	40	34.75	4.25	39.00	18	14	32	8.75

Figure 7 - screen presented after selecting "Fine Arts & Humanities" in figure 6

At this point we have finally arrived at a table containing a list of data at the individual department level. This process continues down to the major level. Using the shortcut links instead of the top level link shaves at most two screens of depth from this process. While this approach allows for more levels of summarization, the excessive number of links which must be drilled through is unwieldy and not particularly attractive or intuitive.

In contrast to the drill down approach presented in the current application, the application designed for this project uses a simple, drop down box driven structure to

provide access to more specific levels of information in less steps, with less choices to be made by the user. Figure 8 shows the application's main menu screen after the user has logged in.

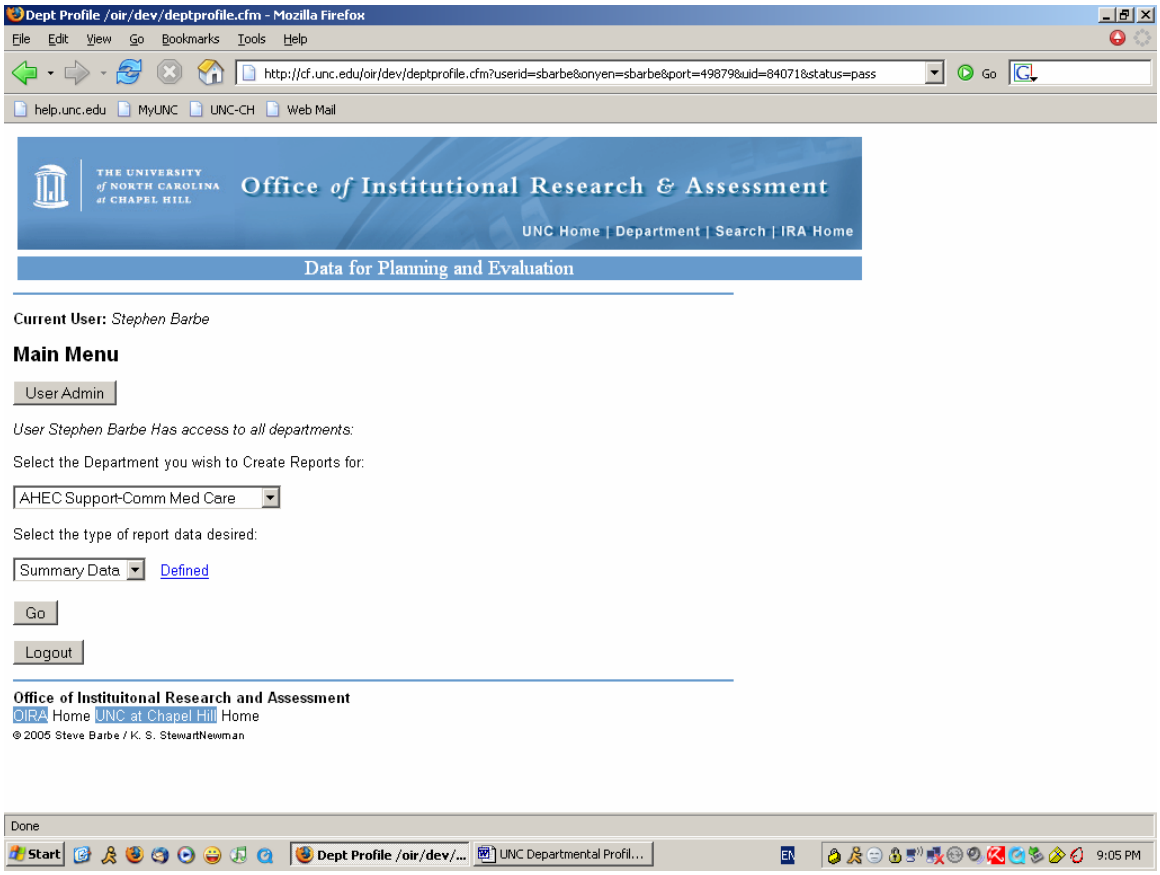


Figure 8 - Project Application Main Menu

The User Admin button is displayed since the user logged in is a System Administrator. This button is not generated for any user who is not an administrator. Beneath the User Admin button are two drop down boxes. The top box allows the user to select a department, while the bottom one allows the user to select summary or detail data. The select box structure of this page allows the user to view a complete, alphabetized list of university departments to which they have access, which gives them a recognizable and

useful place to start their process. Selecting “Art” from the department box and “Summary Data” from the report type box produces the screen in figure 9.

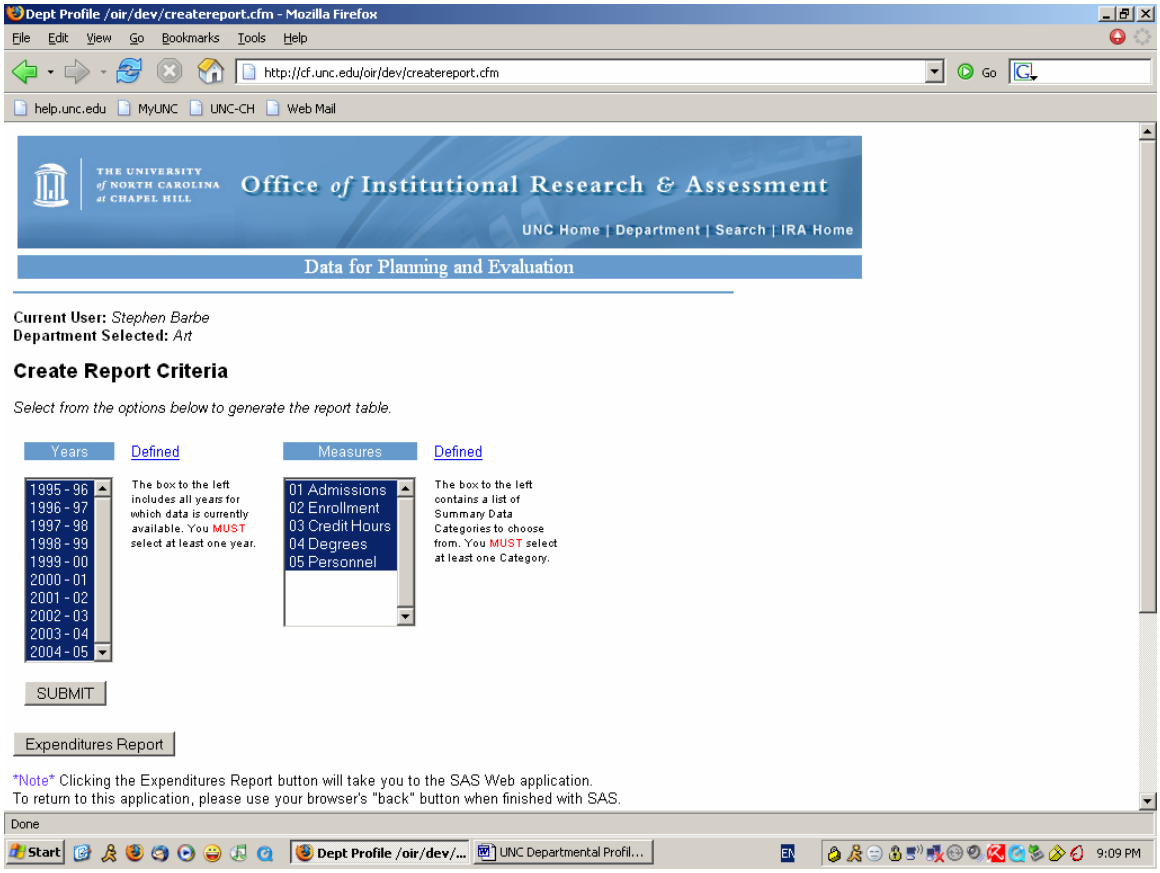
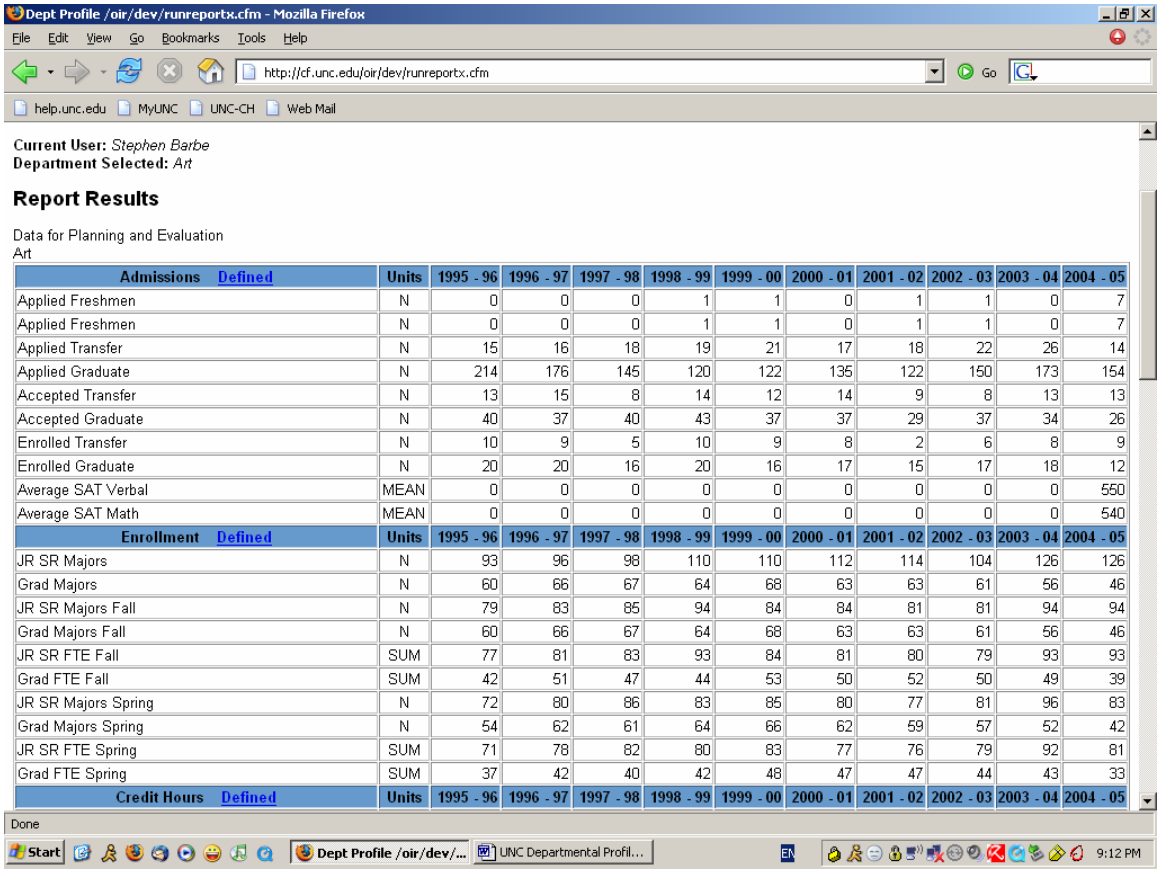


Figure 9 - Screen resulting from the selection of "Art" and "Summary Data"

This screen includes two select boxes; the left one with a list of years for which data is available and the right one listing categories of data available for this department. By default the application selects all years and all measures. Using these boxes, the user may select any or all years and categories of data for the creation of their report. The report is then generated using the selections as query parameters, producing an html table which displays the data for the selected categories separated by headers grouping the

individual metrics by data category. Clicking submit with the default selections provides us with figure 10.



Admissions Defined	Units	1995 - 96	1996 - 97	1997 - 98	1998 - 99	1999 - 00	2000 - 01	2001 - 02	2002 - 03	2003 - 04	2004 - 05
Applied Freshmen	N	0	0	0	1	1	0	1	1	0	7
Applied Freshmen	N	0	0	0	1	1	0	1	1	0	7
Applied Transfer	N	15	16	18	19	21	17	18	22	26	14
Applied Graduate	N	214	176	145	120	122	135	122	150	173	154
Accepted Transfer	N	13	15	8	14	12	14	9	8	13	13
Accepted Graduate	N	40	37	40	43	37	37	29	37	34	26
Enrolled Transfer	N	10	9	5	10	9	8	2	6	8	9
Enrolled Graduate	N	20	20	16	20	16	17	15	17	18	12
Average SAT Verbal	MEAN	0	0	0	0	0	0	0	0	0	550
Average SAT Math	MEAN	0	0	0	0	0	0	0	0	0	540
Enrollment Defined	**Units**	**1995 - 96**	**1996 - 97**	**1997 - 98**	**1998 - 99**	**1999 - 00**	**2000 - 01**	**2001 - 02**	**2002 - 03**	**2003 - 04**	**2004 - 05**
JR SR Majors	N	93	96	98	110	110	112	114	104	126	126
Grad Majors	N	60	66	67	64	68	63	63	61	56	46
JR SR Majors Fall	N	79	83	85	94	84	84	81	81	94	94
Grad Majors Fall	N	60	66	67	64	68	63	63	61	56	46
JR SR FTE Fall	SUM	77	81	83	93	84	81	80	79	93	93
Grad FTE Fall	SUM	42	51	47	44	53	50	52	50	49	39
JR SR Majors Spring	N	72	80	86	83	85	80	77	81	96	83
Grad Majors Spring	N	54	62	61	64	66	62	59	57	52	42
JR SR FTE Spring	SUM	71	78	82	80	83	77	76	79	92	81
Grad FTE Spring	SUM	37	42	40	42	48	47	47	44	43	33
Credit Hours Defined	**Units**	**1995 - 96**	**1996 - 97**	**1997 - 98**	**1998 - 99**	**1999 - 00**	**2000 - 01**	**2001 - 02**	**2002 - 03**	**2003 - 04**	**2004 - 05**

Figure 10 - Report generated with default selections

As we can see, this screen provides us with data for the art department displayed by column years and the metric in the first column of each row. This process does not provide the higher level data summaries available at each level of the hierarchy in the current application, but greatly reduces the amount of work needed for the user to reach the lower level data organized by department. This flattening of the structure becomes particularly useful when the user needs to report data from multiple departments separately. The shortening of the query process pays great dividends in time saved when

generating reports of various types for one hundred or more departments. The design was submitted to the IR&A management in a rough form and approved for use in the prototype application.

With specification 4 satisfied and the need for specification 6 eliminated, the last remaining specification to be implemented for the reporting system was specification 2, which required the system to be able to generate the query result output in a variety of data formats. It was decided that the obvious default method for the display of query results should be to present an html table containing these results to the user. This was a conceptually simple idea, but it was decided that there also needed to be some process for generating table header rows to physically segregate the different categories of report data from each other in order to improve readability. This lead to the creation of 5 initial categories of data to be used for testing: Admissions, Enrollment, Credit Hours, Degrees and Personnel. These categories were used to group related metrics into easily selected options for use in query construction. The data rows stored in the departmental summary data table were then assigned a type ID consisting of a number and a letter. The number designates the category to which that individual metric belongs (1 for admissions, 2 for enrollment, etc.) while the letter is used to separate distinct metrics from each other. For example, the metric “Applied Freshman” (a measure of the number of freshman students who applied to a given department in a given year) has a type ID of 1a, while the metric “Applied Transfer” (essentially the same as Applied Freshman, but applying to Transfer students) has a type ID of 1b. The numeric portion of this type ID was then used to generate a header row based on the change of this value. When the table is constructed, a variable with an initial value of zero is read, and if the value of this variable does not

match the numeric portion of the type ID, a header is generated and the variable is reset to the value of the type id. Whenever the type ID of the data being displayed changes again, another header is generated and the variable reset until all of the retrieved rows have been displayed. In the prototype, the code used to generate these headers was hard coded based on the value of the type ID. While this worked in the way it was supposed to, it made the header generation design non-dynamic in some important ways and would lead to future difficulty if new categories of data ever needed to be added. This was a limitation that was acceptable in the prototype, but was slated for redesign in the beta and production stage.

Upon investigating mechanisms for converting report data into other formats, it was found that the Blue Dragon server did not support any built in mechanisms for creating any of the formats we desired. In order to comply with specification 2, Kay researched the possibility of using a ColdFusion custom tag to provide this functionality. She located several tags authored by different ColdFusion developers and tested them for integration and functionality. After several false starts, a tag was found which enabled the creation of a Microsoft Excel spreadsheet. Installing the tag for use with the application required us to request a configuration change on the Blue Dragon server to enable the tag and allow the writing of a temporary file to the server's hard drive in order to enable the excel spreadsheet to be opened by the user without the need for them to first save the spreadsheet to their local computer. This provided us with a temporary problem in which we needed to conduct a periodic manual cleanup of the server in order to eliminate the Excel files that had been generated by the tag. This problem was deemed acceptable for the prototype phase, but was added to the list of redesign elements to be

solved in the beta and production versions of the application. By the time the generation of the Excel spreadsheet had been enabled, we were very close to the end of the prototype design phase. It was decided at this point that the need for a CSV format for the data was moot, given the ease with which a CSV can be generated from an Excel spreadsheet. The need for a plain text version was deemed similarly unnecessary, and the email option, though still desirable, was scrapped due to lack of time.

When the development of the two application subsystems was nearing completion, the time came to integrate what until this point had been two almost totally separate development efforts into one unified application. Fortunately, several aspects of the ColdFusion development framework made this much easier than we had anticipated.

The ColdFusion application framework provides two special files which can be used to help maintain a unified look and feel to the application, store variables and values that will be true for the entire application, and apply header and footer code automatically to all pages within the application. These two files are called Application.cfm and OnRequestEnd.cfm. The Application.cfm allows the definition of what are called "Application Variables" within the ColdFusion environment. The Application Variables allow the developer to easily define such things as the default data source to query, the path to the root folder containing the application and the timeout settings for the application. In addition to setting the values for Application Variables, the application.cfm file is loaded first any time any other application page is loaded. This allows the inclusion of the page header images, user authentication code module and a standard css style sheet for the entire application in one file instead of being forced to include them or call them from each individual page within the application. This makes

the authentication structure far easier to implement and allows us to functionally generate a “template” for all the pages using headers and footers loaded automatically from only one source. Similar to the application.cfm file, the OnRequestEnd.cfm is loaded automatically after each page is loaded. This allows the definition of one set of footer code for use by all of the pages as well a including a single logout module which displays a button enabling the logout process from each page within the structure.

By making use of the capabilities of these two special files, the process of integrating the two separate parts of the application was made quite simple. Prior to the subsystem integration, Kay and I had been developing code modules completely separately, using separate copies of the Oracle database and separate file storage areas. Since the subsystems were built in a modular fashion, the integration was accomplished mainly through the reconciliation of the two different sets of application variables. We chose one copy of the database to be the actual prototype data source, chose one of the two existing style sheets to be used, defined the application path, and consolidated all of the separate subsystem files in this one location. After integrating the contents of the two original application.cfm files and the original onrequestend.cfm files, the only other integration task that was left to be performed was the creation of a unified Main Menu page for the application. This was very easily done by simply authoring a new menu page which included simple buttons linking to the separate subsystems. Once either the User Administration or the Create Report function is selected from the Main Menu, the appropriate subsystem code is executed with almost no impact at all on the code for the other subsystem. The final step in completing the integration took the form of creating a

series of modules which provided buttons to navigate back to the Main Menu from any point in either subsystem.

During this phase of the prototype development, it was also decided that functional specification 7 (see section III) was made almost completely superfluous due to the elimination of specification 6. It was felt that the only necessary portion of specification 7 in the face of the changes already made to the system architecture was the ability for users to give feedback to the System Administrators from within the system. In order to facilitate this, a simple html mailto tag was included in the onrequestend.cfm file allowing users to send email to either member of the project team from any page within the application.

Once the integration was complete, a very basic round of functionality testing was performed to make sure that the integration had not broken any of the functions of the separate subsystems. When the functionality of the system was verified to our satisfaction, I went through the application documenting all of the possible user options and their uses in order to produce a simple User's Guide to facilitate the instruction of system users (See Appendix II).

VI. Beta Development and Server Migration

The design of the production system also involved several challenges that had not been present in the prototype system. For development of the Production system, we were required to migrate the prototype application from the free server technology we had been using (Blue Dragon) to the Macromedia ColdFusion server, which required some modification of the application code. We were also required to move both the

database and the Web application from a Windows platform to a Unix/Linux platform. Additional functional requirements were added for additional data output formats (PDF) and visual charting of the report information. The User Administration system was also redesigned and the access levels and permissions changed to conform to a new set of user roles and definitions.

The first significant challenge in preparing the prototype application for the beta development phase involved migrating the entire application from the test servers we had been using for development to the servers which would host the final production application. This first necessitated the re-creation of the application database on a different Oracle system installed on a server using the Unix operating system instead of the Windows operating system. The most immediate effect of this phase of the migration was the sudden realization that all of the table and field names contained in the database were now case-sensitive, which they were not when stored using the Windows file system. This required us to modify all of the existing queries used with the application to restore their proper functionality.

The second phase of the prototype migration involved copying all of the application's ColdFusion pages from a Windows Server running the Blue Dragon Server to a Linux server running Macromedia's ColdFusion 6.1 server. After moving the files to the new server, the first thing that we noticed was that the application.cfm file and the onrequestend.cfm file no longer seemed to function at all, which completely destroyed the functionality of the authentication system, prevented access to the application data source and severed the application's link to the style sheets used to format it's appearance. Several hours of attempting to modify these two files failed to restore even a

hint of functionality, but another hour or so of research on the Internet provided the answer to the dilemma. Just as the migration of the Oracle database a Unix file system had made the database case-sensitive, the migration of the ColdFusion pages to a Linux server had done the same to all of the code contained within those pages. When creating a ColdFusion application on a server whose file system is case sensitive, the application.cfm and onrequestend.cfm files MUST be assigned names with the proper capitalization in order for the server to recognize and process them correctly. Changing the "application.cfm" and "onrequestend.cfm" to "Application.cfm" and OnRequestEnd.cfm" respectively restored their proper functionality and allowed us to proceed the with beta phase of the ColdFusion coding.

Several other minor issues were evident in the migration from the Blue Dragon server to the Macromedia server, including some minor syntax differences, very confusing problems with single and double quotation marks, and the cessation of the function of our Excel custom tag.

Once again, the case sensitive nature of the new file system for the Macromedia server necessitated the modification and testing of various portions of the ColdFusion code to correct capitalization errors in modules which referenced files or variables which were suddenly made case sensitive. Perhaps the most confusing and difficult portion of this migration was a change in the way single and double quotation marks are read and processed between the two systems. This seemingly simple and insignificant problem cost us many hours of development effort to correct given the necessity of quotation marks in almost all of the queries and a majority of the function calls used within the system. This issue is sometimes so complex that I was never able to formulate a clear set

of rules for the proper use of either type of quotation marks within ColdFusion. The non-functionality of the Excel custom tag following the migration was an issue that required much more extensive investigation, and will be discussed in the context of the changing requirements included in the beta development plan.

The next challenge in the beta development phase of the project involved the redefinition of the system user roles and the redesign of the user administration subsystem. The management of IR&A decided after initial testing of the prototype system that the inclusion of two different levels of administrative privileges was both unnecessary and confusing. They also identified the need for a role which would have access to all of the data within the system for reporting purposes but needed no access to the user administration subsystem. In order to meet these modified requirements, the existing roles were redefined to include the System Administrator role, the standard User role and a new role called Super User, which replaced the prototype Departmental Administrator role.

The System Administrator role remains largely unchanged in the beta development plan, still allowing the Administrator to make full use of both the user administration subsystem and the reporting subsystem. The User role is similarly unchanged, still requiring that the user be granted access to each separate department and allowing access to reporting data for only those departments. The Super User role, however, is a clear departure from the prototype's Departmental Administrator. The Super User role has access to all university departments for using the reporting subsystem, but has no access to the user administration subsystem. This change was made for several logical business reasons. First, the IR&A management felt that limiting

the System Administrator role to a select few of their staff members would significantly reduce the administrative overhead and possible errors which might be created by allowing multiple, independent Departmental Users to control user access to the application. In addition to the possible administrative problems which untrained administrators might create within an unfamiliar system, several people were identified who have valid reasons to access and report information from any department within the university. It was decided that it would be much easier to establish a role which was granted this level of access by default instead of adding hundreds of entries to the deptusers table for every person who might need this access. These role changes were accomplished with a simple redesign to the deptusers table and a modification of the ColdFusion code which retrieves this access level information and uses it to generate the select boxes which present the users of the system with the reporting options available to them.

Following the redefinition and redesign of the user roles, it was requested that some additional changes to the user administration subsystem be made. The university has several divisions which form distinct organizational units in certain areas which include multiple departments under one nominal heading which applies to the various employees, policies and information pertaining to those divisions. A good example of this is the UNC Chapel Hill School of Medicine. The School of Medicine include 57 individual departments which each have their own staff, equipment and data, but are also all part of the School of Medicine. Instead of requiring a System Administrator to manually grant access to 57 different departments whenever a User is added for the School of Medicine, a new administrative option was created which allows an

administrator to add all of these departments by selecting a single value from a drop down box. This functionality is referred to within the application as “Granting Special Access” and is provided for the College of Arts & Sciences, the School of Dentistry and the School of Public Health in addition to the School of Medicine. In order to fulfill this requirement, it was required that lists of the individual departments under each division be acquired and that these department lists be hard coded into the insert and update queries used to enter this data into the deptusers table. This is one of the few areas of the application which includes non-dynamic queries and has been identified as an area to revisit in a future round of redesigns.

The final challenge in the beta phase was the implementation of multiple alternate query display formats which had been abandoned or not fully implemented in the prototype. In addition to repairing the Excel functionality, it was decided that a method for generating reports in PDF format and the ability to graph the results of the queries needed to be added.

In order to restore the Excel functionality, I located a different custom tag which, while requiring some strange modifications of the query result set in order to correctly pass them to the custom tag, ended up performing better than the tag that had been used in the prototype. Unlike the tag used in the prototype, this tag (called simply CF_Excel) processes the spreadsheet without needing to write a file to the server first. This tag allowed the system to pass the results of the report query directly to the custom tag, but required modification of the custom tag code itself in order to properly display the table header labels and to implement the drawing of empty data cells to preserve the proper column alignment of the report data. I was able to decipher the code of the custom to the

necessary extent to make these modifications. Upon successful testing of these modifications, I contacted the author of the original custom tag, Anujit Ghosh, and provided him with an annotated copy of my modifications to his code. As of this time, I do not know whether he has performed his own round of testing on the modified tag or whether he has made it available for download.

Enabling the creation of PDF files from the reporting results was relatively straightforward in most ways, but did require the creation of a file on the ColdFusion server in much the same way as the original Excel custom tag did. In order to solve this issue, I used one of the existing session variables of the application to generate the name of the file to be stored on the server. In order to automate the cleanup of these files after the User has finished with them, it was necessary to create some additional code for inclusion in the OnRequestEnd.cfm file. This code checks for the existence of a PDF file with a specific name on the server every time a new page is loaded, and deletes this file if it is currently present and not in use. This turned out to be quite an elegant solution for preventing the application server from filling up with useless PDF files. The one issue still unresolved with respect to the PDF creation function within the application is that the custom tag currently being used to generate this output is a free trial version of a commercial custom tag authored by a software development company which engages in professional ColdFusion development. This trial version of the tag does not allow the PDF files that it generates to be printed, but does allow them to be saved to the user's computer with a second page included in the PDF identifying this output as "trial version". The management of the IR&A department has thus far put off the purchase of this tag due to the fact the latest version of the Macromedia server includes built in tags

for generating both PDF files and Excel spreadsheets. This upgrade has been requested, but still has not been approved by the organization responsible for managing these servers.

The final major change in requirements for the beta phase was the implementation of ColdFusion's charting functionality. ColdFusion includes tags which allow query results to be visually graphed using a variety of object formats and chart types. By passing the results of any query to the cfchart tag (and related sub-tags) and choosing which result values to display on each axis, simple two-dimensional charts can be created using a standard x and y axis system.

The formats in which charts may be created include Flash object, JPG image and PNG image. Flash Objects use Macromedia's Flash player technology to create dynamic charts which offer limited interactivity to the user. Flash Charts include dynamic pop ups which display the value of individual data points within the graph when the user hovers the mouse over them, the ability to zoom the chart view in and out, and the ability to print the charts. The drawbacks of Flash charts include the inability to save them outside of the application and the fact that they expire after a certain period of time and must be reloaded.

The JPG and PNG image options are functionally the same, producing charts in standard image formats. These charts can be saved to individual image files, printed, or cut and pasted into other documents. They offer none of the dynamic features of the flash charts, and appear to be of somewhat lower resolution overall.

Once the Chart format has been selected, the chart type may also be chosen. The cfchart tag offers quite a large list of chart types, but for the purposes of this application

only five were chosen. The application provides a drop down box which allows the creation of a line graph, a curve graph, an area graph, a bar chart, or a scatter graph. The way the charting tags currently work, you must select one format and one type for all of the data to be charted, and each result row must be graphed on a separate chart. This is a limitation that we hope will also be solved with the eventual upgrade of the ColdFusion server.

VII. Testing

At this time, the project has been through only one formal round of testing. The system is still in development and additional specifications have been submitted to me for integration into the system, taking us into the third round of development with expanded requirements prior to a second round of application testing.

Throughout the development of the prototype, the project team performed a continuous process of normal developmental function testing to insure that the application code was functioning properly and performing the tasks it was designed for. This process included normal testing methods such as the creation of static queries to provide measures of the accuracy of dynamically generated queries, and intentional misuse of the system to test reminders and error message responses. This process allowed us to eliminate many initial errors in query processing and report presentation without the need of resorting to outside testers.

After the integration phase of the initial prototype construction, the project team members implemented two sessions in which each developer tested the function of the other developer's application code and reported errors which needed to be fixed. Each of

these sessions led to the identification and elimination of several bugs and code inconsistencies that were present in the prototype. Upon completion of these rounds of prototype testing, the application code was submitted to the IR&A management, which resulted in many of the specification changes detailed during the discussion of the beta development phase.

The current status of application testing is on hold pending completion of the third round of development and integration of new functional specifications which are outside the scope of the current paper. Formal testing will resume upon completion of the current development phase.

VIII. Deployment

As of the writing of this paper, the application has not yet reached the deployment phase. During the continued development of the application, it has thus far been distributed only to those IR&A staff who have been involved in the development and testing of it. The Management of IR&A was involved in the first round of formal testing and had many positive comments as well as a few negative ones.

The most salient comment to come from the first round of formal testing was the need for a more complete and up to date User's Guide focusing on the use of the reporting subsystem instead of the function of the user administration subsystem. Since almost none of the application's daily users will be administrators, the new User's Guide includes almost no discussion of the system administration role and focuses expressly on outlining the reporting subsystem's process flow and capabilities. This was deemed a

crucial requirement to be met prior to the consideration of even limited application deployment.

In addition to the revision of the User's Guide, it was suggested that there should be more user instruction included within the application itself. This suggestion lead to the creation of many on screen text boxes providing short summaries or instructions about various page features. There are also several links to html version of the system User's Guide, including anchor links to various specific definitions or instructions related to the currently displayed screen options. The newest revision of the User's Guide is included in Appendix III.

Deployment plans are not currently proceeding, but some members of the IR&A staff have begun the process of planning for limited beta testing deployment which will be used as both a third round of formal testing and the statistical basis for more focused capacity planning and resource requests. This process is unlikely to begin until the spring of 2006.

VIIII. Conclusion:

This project has resulted in the development of a web application intended to provide a more flexible and intuitive system for reporting the data gather by the Institutional Research and Assessment department of the University of North Carolina. Throughout the envisioning, planning, design and development phases of the project, the goals and specifications of the system have continued to change and have required constant updates to the project plan. The planning and implementation of the project had at all times loosely followed the four phases of the Software Development Life Cycle

model. This model provided a great deal of guidance in maintaining the conceptual integrity of the final product.

The Envisioning phase began with the gathering of the intended end user's goals, requirements and expectations for the capacities of the completed system. The data provided by the envisioning process lead directly into the formulation of the business requirements and functional specifications used to guide the development of the final application. Prioritization of these requirements allowed the project team members to clearly identify the most important aspects of system function and ensure that these requirements were met prior to work being started on several of the lower priority aspects of the system. The loss of a team member and lack of time at the end of the development phase made clear to the remaining team members the importance of this level of focus in achieving final results that were deemed acceptable by all concerned parties even though some of the specifications were not met in the initial prototype design and construction.

The transition between the prototype phase and the beta phase clearly illustrated the cyclical nature of the SDLC by requiring immediate re-evaluation of the project vision and subsequent modifications to the functional specifications. The migration from the prototype environment to the production environment also provided the project team with valuable experience in the identification and resolution of cross-platform software development issues.

The near-continuous re-evaluation and modification of the project goals also served to illustrate one of the most prevalent problems in software development, "feature creep". Had the team been able to adhere strictly to the initial specifications and resist the need to expand the application's functionality, it seems clear that the application

could have been put through sufficient testing to have begun production use of the system several months ago. As of the writing of this paper, the project team has begun to question whether the application will ever be completed, and if it is, whether the original vision of the project will be preserved in the final results.

This project ended up producing a Web Application which is similar in many ways to other web reporting and query systems in use today. The use of standard html form elements to pass data to the dynamic portions of the code and the design concept of using drop down and select boxes to provide limited, well-defined user input options to tailor their report queries are widely used and accepted. Indeed, studies of other reporting systems of this type informed many of out interface and query design decisions. Many of the challenges faced and solved in the session continuity and query design areas have provided the project team members with excellent solutions which have since been used in the design and development of other web query interfaces. Perhaps the most unique aspect of this project was the aforementioned strangeness of the main report data table, named dept_summ_dm in the final Oracle database. The pre-summarized nature of the data allowed us to be certain that there would need to be no mathematical functions applied to the data stored in the system under any circumstances. The fact that the system is meant to provide a historical reporting tool meant that we were able to use the years in which the data was collected as database columns instead of individual cell data within the database records. Because each metric is always collected for every year and the metrics reported needed to be presented in a row form instead of a column form, this table ended up having its rows and columns functionally “flipped” from what the normal attribute design would look like. This unique arrangement actually ended up simplifying

the construction of the query engine and formatting code a great deal due to the fact that a normal table arrangement would have required that the results be pivoted in order to present them in the desired configuration on screen.

Another interesting difference in this project compared to others that I have worked on is the use of the UNC onyen system to provide a unified login and authorization model for the application. In many other systems of this sort that I have worked with, the application is designed to use either its own login and access control system or one provided by the network operating system. The use of a separate script to provide authorization through a unified "third party" system was entirely new to the project team. The benefits of this system became immediately apparent when the project team realized that we were not responsible for the security or policies in place in the onyen system, which decreased the prospective administrative workload for the User Administration subsystem tremendously. It is also very convenient to have access to the application controlled by the same user ID and password that the end user uses to gain access to both their workstation and the UNC campus network. This portability of and ease use made a dramatic impression on the project team members.

Regardless of the perils revealed in the project management and envisioning areas, there are several features of the application that we feel have been highly successful, and some that we believe did not succeed as well as we would like.

Perhaps the most successful feature incorporated into the design philosophy of this project is the modularity of the code. Every effort was made to ensure that the final application code was as dynamic as possible in nature. The use of session variables to keep track of user information and choices greatly reduced the amount of effort required

by the users to learn and operate the system. This also allowed all of the query processing and user authentication operations to be automated in such a way that addition and removal of data from the system has little to no effect on the functioning of the system. There are still a few instances of non-dynamic user and system interaction which the project team hopes to automate and modularize at the earliest opportunity, but the overall flexibility and power of the application has been a somewhat pleasant surprise.

Another notable success in the application design was the modification of the user administration system in the beta phase. This restructuring helped to solve several administration and business logic problems before they ever really had an opportunity to materialize. The nature of the data provided by IR&A allowed an extremely high amount of administrative streamlining by limiting manipulation of users and data to a very small number of administrative users who are intimately familiar with the system and capable of handling all of the maintenance and updating of the system.

The query construction dialogs and options also allowed for far more simplification of the user interface than was initially predicted. We believe that this will make the final application even more attractive and usable for the eventual end users of the system.

Of the features that were not as successful as we hoped, the primary ones have to do with the alternate formatting of report data. While we managed to enable the Excel, PDF and charting functions of the application, they are not as efficient or flexible as we would like. One of the most frustrating causes for this issue is our lack of ability to configure and control our own server environment. Research has shown that many of the problems encountered in the formatting arena would be significantly reduced by the

simple expedient of upgrading the software version of the ColdFusion application server. Unfortunately, this is something that the project team does not have the resources or authority to do.

Another feature that we feel needs further investigation and modification is the underlying structure of the database tables and records. It is currently not possible to identify with a sufficient degree of accuracy which metrics exist for which departments. The create report dialog is currently filled with a massive number of non-academic departments which have no data aside from personnel information and include no detail data which can be retrieved with the current reporting queries. This problem is also currently under investigation and a solution will hopefully be found.

Finally, perhaps the most minor issue concerning the project team is the relatively unattractive and out of place feel of the initial user authentication mechanism. Currently the Perl script supplied by the UNC onyen group is the only known way to implement the use of the onyen's unified login capabilities. It is currently hoped that the team may gain access to an experienced Perl scripter who can bring this mechanism more in line with the presentation of the rest of the interface.

Perhaps the most important lesson learned during the course of this project is the problem of feature creep which has been discussed previously. As a developer, I have begun to be more cautious about acknowledging the ability to modify the system in certain ways merely to cut down on the amount of changes which seem to be continually required. This is especially true in the context of this project due to my extremely limited amount of influence over the continuing vision of what capabilities of the system are actually necessary for its proper operation. I would recommend that developers provide

as much input as possible into maintaining the structure and integrity of the functional specifications for their projects in order to hopefully make those they are accountable to aware of the problems that continually expanding requirements present

Another very important lesson learned due to the length of time over which this project has been conducted is the importance of well-commented application code. Being forced to return over and over again to code which was thought to be complete and frozen months previously has made the value of clear and comprehensive comments absolutely plain. Good commenting and explanation has saved the developers countless hours of attempting to parse and understand the code that they have already written when changes to the system have been required. Even though I have not begun commenting while in the actual process of coding, I have made it a policy to allow no more than two days to pass between my completion of a code section and a revisit in order to comment that code. I would recommend some standardized commenting practice to anyone involved in application development.

In addition to value of informative comments, the project team also learned the advantages of modular code design and reuse. We recommend that one of the primary areas to be considered in the development strategy of any application is an analysis of process areas which will lend themselves to repetition. These sorts of tasks (providing a constant display of who is logged in, generating the same list of options on multiple pages, etc.) provide the basis for mapping out specific portions of the application code which only really needs to be written once, and can then be called or referenced whenever needed. Again, this is not realty a revelation, but the practical experience of the project team has proved once again the value of this development ideal.

Finally, the project team has become aware of the fact that a good dynamic web application begins with good database design. A thorough knowledge of the characteristics and purpose of the data contained in the database makes it possible to design a database that is easy to use for specific purposes which are defined in the requirements of the system. This sort of knowledge convinced the project team to design a database table for this application which would likely be totally unusable for many other purposes. The simplicity of the database portion of the application has made modification and tuning of queries much easier than was originally foreseen, unfortunately, it has also made attempts to expand the variety of data stored in the database somewhat problematic. Continued desire for enhanced system functionality has recently been expanded to include possible redesign of the underlying database structure, and the project team is currently unsure how the report query engine will respond to these structural changes.

The future of this project is currently in flux, but it will surely proceed into areas that were not accounted for in the original vision. Whatever direction future development takes, however, the project team believes that modular, dynamic program code will continue to feature heavily in the goals of the project. The system also contains plenty of opportunities for query optimization and performance enhancement. As the live data sets for the application continue expand, performance will most likely become a driving force in future development.

Bibliography

Chigrik, Alexander. “A Comparison of SQL Server 2000 with Access 2000.” Database Journal. 14 May 2003. Jupitermedia. 9 Nov. 2005 http://www.databasejournal.com/features/mssql/article.php/2204341.

Foch, Craig B. Oracle9*i* Database Administrator's Guide, Release 2 (9.2) for Windows. Oracle. 2002. Oracle. 9 Nov. 2005 http://download-west.oracle.com/docs/cd/B10501_01/win.920/a95491/specs.htm#1006174.

Hojtsy, Gabor. “PHP Manual”. PHP. 28 Oct. 2005. The PHP Documentation Group. 11 Nov. 2005 http://www.php.net/manual/en/introduction.php.

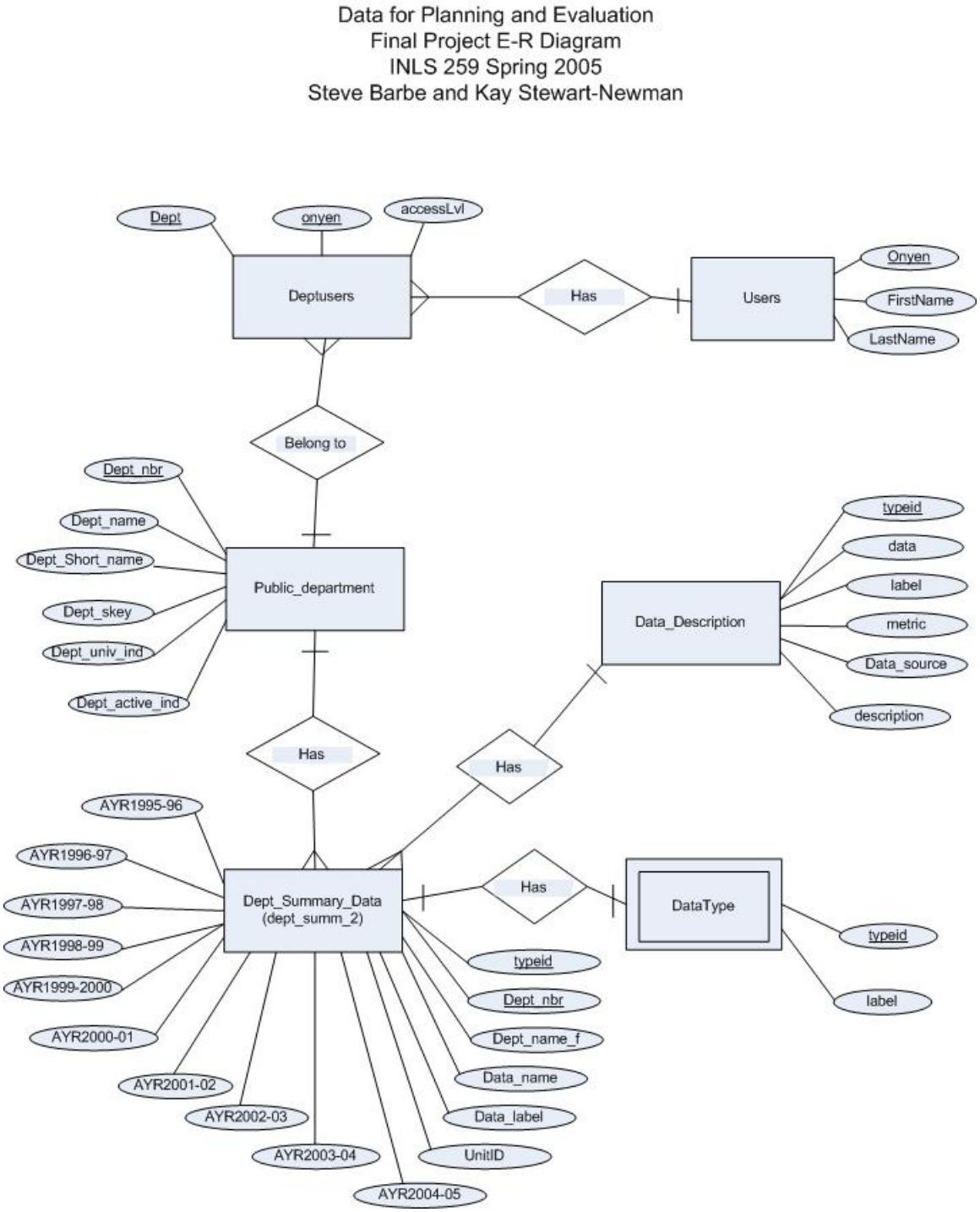
“ASP Overview.” Microsoft. 2005. Microsoft. 11 Nov. 2005 http://msdn.microsoft.com/library/default.asp?url=/library/en-us/iissdk/html/526bf565-7f3e-4c60-a78b-bdac64ebb0e0.asp.

“ColdFusion MX 7 FAQ.” Macromedia. 2005. Macromedia, Inc. 11 Nov. 2005 http://www.macromedia.com/software/coldfusion/productinfo/faq/#item-1-6.

"OLAP Cube." Wikipedia. 15 Nov. 2005. Wikipedia. 11 Nov. 2005 http://en.wikipedia.org/wiki/OLAP_cube.

"Top Ten Reasons to Use MySQL." MySQL. 2005. MySQL AB. 11 Nov. 2005 http://www.mysql.com/why-mysql/toptenreasons.html.

Appendix I: Database E-R Diagram



Appendix II: Prototype User's Guide

Data for Planning and Evaluation
System Users Guide

Purpose:
This document is a short tutorial on the structure and usage of the Data for Planning and Evaluation Web Reporting System. It contains an explanation of the performance of some basic system tasks and basic information about the structure and organization of the system.

A word about access levels:
This application supports 3 levels of user access to the system: System Administrator, Department Administrator, and User. The Administrator access levels allow for administration of user access to the system and the various departments for which this application contains report data. Descriptions of the rights associated with these access levels is as follows:

- System Admin – A system admin has complete access to user administration for the system. System Admins can add and Remove System Admins, Department Admins and users from the system. They also have the ability to add and remove access to departments, and may modify the existing access levels that users have to departments
- Department Admin – A Department Admin has limited access to user administration for the system. A Department Admin may add and remove a user's access **only** to the department for which they are a Department Admin. They may also change the current access level to that department.
- User – A User has access only to the reporting functions of the system. A user may have access to data for one or more departments, but can create reports for only one department at a time.

End Session:
The End Session button allows you to end your application session and log out of the system. It will appear at the bottom of most pages in the application allowing you to logout at virtually any time.

I. Starting the Application:
Click or cut and paste the following link in your browser to start the application:

http://takahe.ils.unc.edu/projects/inls259/sbarbe/Project/deptprofile.cfm

The first thing you will see is the application login page. Enter your UNC Onyen and password, then click continue to log in to the system. This will take you to the main menu if you have access to the system for more than one dept (see **II. Main menu**) or to the task selection page if you have access for only one department (see **III. Task Selection**)

II. Main Menu

The application Main Menu appears if you have system access to more than one department. The next step in accessing the reporting system is to select the department you wish to administer or create a report for. After selecting the department from the drop down box, click the **Go** button to continue to the task selection page.

III. Task Selection Page

Based on your level of access, you will see either one or two task selection buttons on this page, as well as buttons at the bottom of the page which allow you to return to the Main Menu or End Session and logout of the application.

All users will have a **Create Report** Button which allows them to select Summary Data for planning and evaluation from the department chosen from the main menu (or the department they have access to if they have permissions for only one department) Clicking the **Create Report** button will take you to the Create Report Page (see IV. Create Report Page)

Administrators (both System and Department) will have a **User Administration** button in addition to the Create Report button. Clicking the **User Administration** button will take you to the User Administration Page (See V. User Administration Page)

IV. Create Report Page

This page allows the user to create a report to pull data for planning and evaluation from the Summary Data Database. To create your report, do the following:

1. Select from the radio buttons to choose how you wish your data to be displayed
2. Select the years for which you wish to retrieve data from the select box on the left. You can hold down the shift or ctrl key and click the list items to select multiple years for display. The past five years are selected by default.
3. Select the measures you wish to include in your report from the select box on the right. Again you may use shift or ctrl and click to select multiple measures. The first measure in the list is selected by default.
4. Click the **Submit** button to run your report. This will submit your report to the database and return the results of the report to you. Once your report has been run, the resulting page will include buttons allowing you to **Select a Department** (returning you to the Main Menu) or **Create a report** (which returns you to the Create Report Page).

V. User Administration Page

This page contains 3 buttons for user administration functions which will allow the user to perform administration tasks on a system wide or departmental level based on their access level to the department selected from the main menu.

V.1 System Administrator

A system administrator may choose from the following 3 tasks:

- **V.1.1 Add User** – clicking this button will take you to the add user form. This form requires you to enter the first name, last name, and onyen for a

new system user in the text boxes provided. You may then select the department for which the new user will have access, and the access level they will have. Clicking the **Add User** button will add the user to the database with the selected access level to the selected department. If you wish to give a user access to multiple departments, you will need to modify the user after adding them (see V.1.3 Modify User). This series of pages also include buttons at the bottom which allow you to jump back to earlier pages in the application.

- **V.1.2 Delete User** – Clicking this button will take you to the delete user form. *CAUTION* Deleting a user from the System Admin version of this page will remove their access to the Application entirely! If you wish to remove access to only one department, see (V.3 Modify User). The Delete user form at the System Administrator level provides a list of all of the users currently authorized to use the system. To delete users from the system, select the check boxes next to their user information and click the **Delete User** button *CAUTION* there is at this time no request to confirm deletion. When this button is clicked the records will be deleted, so only click the button if you are sure you wish to remove the selected users. After clicking the delete user button, the results of the deletion will be displayed on the next page. This page also includes buttons to return to the delete user form (**Delete More Users)** as well as buttons to jump back to earlier pages in the application.
- **V.1.3 Modify User** – Clicking this button will take you to the Modify User Lookup page. On this page, select the user you wish to modify from the drop down box at the top and click **Lookup User**. This will retrieve the user's current information and take you to the Select Modify User Task page. This page includes the following button:
    - **V.1.3.1 Update User Information –** Clicking this button will take you to the update user name form. On this form, you may change the first or last name of the user by entering the new values in the text fields provided and clicking **Update User.** This will commit the changes to the system and display the new user name information. This page also includes buttons to return to earlier pages in the Modify User and User Administration trees.
    - **V.1.3.2 Change Existing Access Levels** – This button takes you to the Change Access Level Form. This form displays all departments to which the current user has access and the access levels to those departments. You may modify the access level to only one department at a time by selecting the radio button next to the department to be modified and clicking **Change Existing Dept Access Level**. This will take you to the Change Access Level Page. This page displays the current department selected and the user's current access level. Select the new access level from the drop down box and click **Update Dept Access Level** to change the user's access level to the selected department. Once again these pages include buttons that allow you to jump back to earlier pages.

- **V.1.3.3 Add New Department Access** – Clicking this button takes you to the Add New Department Access Form. This form displays the name of the user currently being modified and allows you to select a new department to provide access to and a new access level for that department. Select the appropriate values and click **Add New Department Access** to add the new access.
- **Delete Department Access** – Clicking this button takes you to the Delete Department Access Form. This form displays the name of the current user being modified and a list of all of the departments to which they have access. Select the check boxes next to any departments for which you wish to remove access and click **Remove Existing Department Access**. *CAUTION* As with delete user, there is currently no provision to confirm your delete selection. Only click this button if you are sure you wish to remove this access. Buttons on this page also allow return to previous pages.

V.2 Department Administrator

A department administrator has access to many of the same functions as a system administrator, but may only perform them regarding users in the department that they are an administrator for. A department administrator may choose from the following 3 tasks:

- **V.2.1 Add User** – The Add User form is very similar to the form provided to system administrators (see V.1.1) except that the department is automatically assigned and may not be changed.
- **V2.2 Delete User** – Clicking this button takes you to the Department Delete User Form. This form displays a list of all the current users with access to the department. To remove a user's access to the department, select the check box next to the user and click **Delete User**. *CAUTION* As with delete user, there is currently no provision to confirm your delete selection. Only click this button if you are sure you wish to remove this access.
- **V2.3 Modify User** – The Modify User form for department administrators allows only one function, changing the user's access level to the department. Clicking the modify user button takes you to the Modify Department Users Form. This form allows you to choose department users from a drop down box. Selecting a user and clicking **Lookup User** takes you to the Change Department Access Page. This page displays the current user being modified, the department, and the user's current access level. Clicking the **Change Current Access Level** button takes you to the Select New Access Page. This page allows you to choose the new access level from a drop down box. Departmental administrators, however, may only assign user or department administrator access, they may not add system administrators. Selecting the new access level and clicking **Update Dept Access** updates the user's access level to the new value.

VI. Questions or Difficulties:

If you have questions about this document or difficulties with the Application, please contact Stephen Barbe (sbarbe@email.unc.edu) or Kay Stewart-Newman (kaysn@email.unc.edu) via email and we will attempt to address your concerns.

Appendix III: Beta User's Guide

Data for Planning and Evaluation
User's Guide

Purpose:
This document is a short tutorial on the structure and usage of the Data for Planning and Evaluation Web Reporting System. It contains an explanation of the performance of some basic system tasks and basic information about the structure and organization of the system.

A word about access levels:
This application supports 3 levels of user access to the system: System Administrator, Super User, and User. The Administrator access levels allow for administration of user access to the system and the various departments for which this application contains report data. Descriptions of the rights associated with these access levels is as follows:

- System Admin – A system admin has complete access to user administration for the system. System Admins can add and Remove System Admins, Department Admins and users from the system. They also have the ability to add and remove access to departments, and may modify the existing access levels that users have to departments
- A Super User has access only to the reporting functions of the system. A Super User has access to data for all departments, but can create reports for only one department at a time.
- User – A User has access only to the reporting functions of the system. A user may have access to data for one or more departments, but can create reports for only one department at a time.

Logout:
The Logout button allows you to end your application session and log out of the system. It will appear at the bottom of most pages in the application allowing you to logout at virtually any time.

I. Starting the Application:
Click or cut and paste the following link in your browser to start the application:

http://cf.unc.edu/oir/dev/deptprofile.cfm

The first thing you will see is the application login page. Enter your UNC Onyen and password, then click continue to log in to the system. This will take you to the main menu (see **II. Main menu**).
Throughout the application, the name of the current user (almost always you) will be displayed at the top left side of the screen underneath the Application Header. After you have selected a department for reporting, the name of the currently selected department will also be displayed underneath the current user's name.

II. Main Menu

The application Main Menu appears if after you successfully log in to the application. The Main Menu Page includes two select boxes: Department and Report Type. Select the department you wish create a report for and the type of report data desired from these drop down boxes. The report data type drop down box allows you to select either Summary Data or Detail Data:

- Summary Data: This selection gives you access to a number of predefined groups of metrics which are commonly desired for certain reporting purposes. These groups include such selections as Admissions, Enrollment, and Credit Hours which group similar metrics together for convenience.
- Detail Data: This selection allows you to select individual metrics in order to more easily customize your reports to include only the data that is necessary for your purposes. Note: Detail Data is generally available only for Academic departments. Selecting Detail Data for a department for which Detail Data does not exist will result in a message on the Create Report Criteria page stating that Detail Data is unavailable for this department.

After selecting the department and report data type from the drop down box, click the **Go** button to continue to the Create Report Criteria page.

III. Create Report Criteria Page

The Create Report Criteria page includes two select boxes which allow you to choose which data you wish to include in the report that you generate. The left select box is a list of all of the academic years for which data is currently stored in the system. The right select box is list of either data categories or individual metrics based on the report type selected on the main menu. By default, all years and all categories or metrics are pre-selected. If you wish to create smaller reports with a limited set of years or metrics, you can use the **CTRL** key and the mouse to select the years and metrics you want from the boxes. Simply hold down the **CTRL** key and left click all of your selections in each select box. When you have completed your selections, click *Submit* to generate your report. This will process your selections and generate an HTML report based on the years and metrics selected, taking you to the Report Results page where you will also find additional reporting options.

Also on this page you will find a button called *Expenditures Report*. Clicking this button will launch a SAS web Application which displays Expenditures information for the selected department. In order to return to Departmental Profiles from the Expenditures Report, use your browser's back button.

IV. Report Results Page

The Report Results page displays the data that you requested by making your selections on the Create Report Criteria page. The report is generated as an HTML table by default. The table displays the Metrics selected in the leftmost column of the table. The next column, labeled "Units", provides a code which indicates what type of statistic ([N]umber, Mean, Sum, etc.) is used to present the data for that metric. The rest of the columns in the table present the data for the years requested in the report. If Detail Data

was selected for the report, the table will include an additional column labeled "Major" which provides data for each separate major offered by an academic department.
The Report Results Table also includes a set of headers which categorize the individual metrics into the categories used for the Summary Data report. Each of these Headers (Admissions, Enrollment, etc.) includes a hyperlink which leads to a definition of the categories used to organize the reporting metrics.

Immediately below the Report Results table are two buttons: *Create another Report* and *Main Menu*.

- Create Another Report: This button returns you to the Create Report Criteria page, allowing you to choose a different set of years and/or metrics to generate a different report for the department that is currently selected. As always, the current department is displayed at the top left of the application.
- Main Menu: This button returns you to the Application Main Menu.

Beneath these two buttons, you will find a box titled **Alternate Report Formats**. This box includes a set of radio buttons which allow you to either export the report results to an alternate file format (Microsoft Excel or Adobe PDF) or to create charts based on the data returned in the report. To use these features, select the radio button for the desired format and click **Submit**.

- Download as Spreadsheet: Selecting this radio button will produce a box allowing you to either open the report results in Excel, or to save the report results to your computer in Excel Format. Select either: "Open with Excel" or "Save to Disk" and then click *OK*.
- Download PDF: Selecting this radio button will convert the report to PDF format and open the report in your browser. Once the PDF is displayed, you may print the PDF report (Kay – This is currently unavailable until we buy a copy of the CFX_PDF custom tag) or save it to your computer. After generating a PDF report, you will need to use your browser's *Back* button to return to the Department Profiles Application.
  Note: In order to use this option, you must have Adobe Acrobat Reader installed on your computer. If you need this software, you may download it for free at: http://www.adobe.com/products/acrobat/readstep2.html
- Create Chart: Selecting this radio button will take you to the Create Chart Criteria page (see section **V. Creating Charts**).

V. Creating Charts

Selecting "Create Chart" from the Report Results page takes you to the Create Chart Criteria page. This page includes two or three select boxes (two for Summary Reports and three for Detail Reports) and two drop down boxes which are used to select the data to be charted, the format of the chart, and the type of chart desired:

- Years: This is the same type of select box used for the Create Report Criteria page (See **Section III. Create Report Criteria**) and functions in the same way.
- Measures: This box is the same as the one for the Create Report Criteria page and functions in the same way.
- Major (Detail only): If Detail Data was selected for the report, a third select box labeled "Major" will appear on this page. This box displays the majors offered by

the currently selected department and allows you to select which majors you would like to create charts for. It functions as the other two select boxes do. If detail data was selected, you **MUST** select at least one major for creating charts.

- Select Chart Format: This drop down box allows you to choose which format you wish to use for chart creation. There are three options available:
  - Flash Object: This option will produce a chart using Macromedia's Flash technology. You must have the Macromedia Flash player plug-in installed to use this option. Flash charts are dynamic objects which allow several options once the chart has been generated:
    - Moving your mouse pointer over a data point on the chart will cause a supplemental information bar to be displayed, which lists the Metric this chart displays and the value of the data point the mouse pointer is resting on.
    - Right-clicking on the chart brings up a menu which will allow you to expand the chart view in or out or print the chart. The other options in this menu are related to the Flash plug-in configuration, and have no bearing on the Departmental Profiles application.
  - JPG image: This option will produce the charts as JPG image files. These images are static once generated and do not offer any of the Flash Object's dynamic features. Right clicking on a JPG chart will give you the standard image menu provided by your Web Browser. This menu will allow you to copy or save the chart to a JPG file. Once saved as an individual image file, you may print the chart or insert it into other documents as you would any other JPG image.
  - This image is exactly like the JPG image option, except that it generates a PNG (portable Network Graphic) image instead of a JPG.
- Select Chart Type: This drop down box allows you to select the type of chart you wish to create. There are five Chart Type options:
  - Line chart: This option will produce a standard line chart with the charted data points connected by straight lines from one point to the next.
  - Curve Graph: This option will produce a chart where the data points are connected by a smoothed curve line joining the data points together.
  - Area Graph: This option will produce a chart with the data points connected by straight lines (similar to the line chart) and the area below those lines shaded blue.
  - Bar Chart: This option produces a standard bar chart from the data selected.
  - Scatter Graph: This option produces a Chart in which the data points are graphed, but not joined in any manner.

After you have selected the years and measures for the data, the chart format and the chart type, click the "Submit" button to generate your charts. This will take you to the Chart Results page.

The Chart Results page includes three navigation buttons at the top of the page which will allow you to return to the Create Report page, the Create Chart page or the Main Menu. The Logout button is also present.

The Charts are displayed along the left side of the page and a scrollbar will be included on the right hand side of the Browser if the charts extend past the bottom of a single Browser page.

VI. Questions or Difficulties:

If you have questions about this document or difficulties with the Application, please contact Kay StewartNewman (Kay_StewartNewman@unc.edu).