Fall 2006

Creation of Pair Test Online Application

Kevin R. Hayes
Regis University

Regis University
School for Professional Studies Graduate Programs
Final Project/Thesis

Disclaimer

Creation of PAIR Test Online Application

Kevin Hayes

Regis University

School for Professional Studies

Master of Science in Computer Information Technology

Abstract

This project researches, analyzes, designs and implements a software application to provide the ability for the PAIR Test to be automated. Currently, the PAIR Test is only available in an offline/manual format. The offline process is not efficient and the data collected does not get stored into a database. The goal of this project is to enable the PAIR Test to become more efficient and have the ability to store results of the testing process into a database for future data analysis. The system will also remove the manual scoring and relationship profile creation by automating this process, resulting in the automatic creation and population of a Portable Document Format (PDF).

Acknowledgement

I would like to thank my wife and kids for all the sacrifices they made to allow me to complete this project. Cindy, I could not have completed this without your support and patience. Thank you sincerely.

I would also like to thank Gene Mastin for allowing me the flexibility and trust required to build this application. I really have enjoyed working with you and cannot express enough how much patience you showed me during this project. Thanks Gene.

Document History

Date	***Revisions***	***Document***
5/10/2006	***Original***	***First Draft .1***
5/23/2006	***Changed chapter names and elements to standards per Joseph Gerber response***	***Paper Draft .2***
5/29/06	***Chapters 3, 4, 5***	***Drafts***
6/4/06	***Chapter 2 and comments from peers in 696B***	***Draft***
6/8/06	***Changes from Joe Gerber Comments***	***Draft***
6/13/06	***Final draft comments and conclusions***	***Draft***
6/16/06	***Updates from Advisor***	***Draft***
6/24/06	***Final revisions from advisor and instructor***	***Final Draft***

Table of Contents

Chapter 1 - Introduction

This paper documents the lifecycle and development of the PAIR Test website and online test application. PAIR stands for Psychological Audit of Interpersonal Relationships. The PAIR Test is a relationship compatibility inventory used by professional counselors, marriage and family therapists, social workers and pastoral counselors.

Problem Statement

Gene Mastin, PhD is the owner and operator of the PAIR Test. Dr. Mastin has owned the PAIR Test for several years now. When Dr. Mastin purchased the PAIR Test, the test consisted of only the testing booklets, forms to score the results/relationship profile, and the current customers. In 1997, Dr. Mastin contracted to have a website created for the PAIR Test. This site was built to facilitate the ability for counselors wishing to begin implementing the test as part of their practice, and the ability to order these test booklets and other materials via the internet. The site was mainly used by professional counselors and there was only a single login that all counselors used to be able to order the testing products. After several years of growth, the number of counselors has now reached over 2000 currently using this test as a standard practice in the marital and pre-marital counseling practice.

Description of Existing Situation

The PAIR Test is a psychological and relationship inventory tool that is used to assess the compatibility issues for premarital and marital therapy. Prior to this project, this test was not offered in an automated version.

The workflow before the author undertook this project was as follows:

1. Approved counselors logon to www.PAIRTEST.com and complete a purchase request for hard copy forms of the test booklets and profile scoring forms.
2. Once the purchase is successful, the administrator for PAIR Test received an email notifying of the delivery address to ship hard copies of the order to the counselor. (PayPal currently handles payment processing).
3. The counselor then has their respective client/patient complete the test by using the booklets to answer 500 questions with a true or false answer on an answer sheet.
4. The counselor creates the relationship profile form using the answer sheets from both persons that completed the test. This is done manually and is time consuming.

5. The profile is then used as a valuable therapeutic discussion tool during a counseling session with the counselors and the patients/couples.

The scoring procedure is labor intensive and takes several hours of office time for the counselor. The adoption rate of the PAIR Test has been slow due to the manual effort required to complete the test, administer the test, print and update new test booklets, and create the relationship profiles.

After completing the analysis of the existing process and conducting interviews with the owner of PAIR Test, the author felt that the current situation would be feasible to implement the process via an online web based application. The author saw nothing in the requirements or business domain that raised any issues that he felt could not be delivered using either custom proprietary software or possibly even a survey or assessment software package. The initial assessment was that the customer had no experience in this type of application building process and that the author would need to drive and define the requirements, design, implementation and deployment of the final product.

Goals of the Project

The main goals of this project are to eliminate inefficiencies and improve the process for the PAIR Test, re-engineer the user interface and

interactions on the PAIR Test website for the new user types, and provide a way to retain test results in a database. An online/automated application will save significant time for patrons and counselors. An automated online version of the PAIR Test will provide for the following:

- Speed and accuracy of completed test results for couples
- Reduced costs for testing materials and administration
- Attract and retain new counselors
- Access to test results for future data mining and relationship profiling

This thesis focuses on the benefits of creating an online version of the PAIR Test (www.pairtest.com). The current test requires couples to complete a 200 or 500 question test. Next, the counselors have to review, score and complete the relationship profile from the test results. These steps are all completed manually and can take hours of several individual's time. This project provided a web-based system that automates the test, scores the test results, and collects payments for the test. The online version also provides the ability to create, update, and delete user accounts and test questions. The project also provides an auto-generated relationship profile from the data in the database. The success of the project is based on the completion of the application according to the specified plan and to implement the functional scope.

Author's Role in the Project

The author performed many roles for this project. The author acted as a business analyst in gathering and documenting the functional requirements. This included stating the items that were delivered and those that were deferred for future phases of this project. The author also acted as the interactive/web designer for the user interface. The user was the database administrator for the MySQL database. Along with all the other elements of the project, the author provided quality assurance testing and deployment engineering. The business owner (Dr. Gene Mastin), was the domain knowledge expert. He defined the requirements and provided a final decision for what functionality got delivered. Dr. Mastin was not responsible for any of the technical items related to this project. This was the role of the author.

Constraints

The project was constrained by the budget, scope of functionality and technical architecture. The budgetary constraint imposed that the technical architecture use as many open source technologies as possible. There was only one developer, the author. No other resources were brought in to help with any coding, testing, or management of this project. The project was limited to technologies that are Java or Java compatible components, as this was the author's area of expertise. The project plan and timeframe did not allow ample opportunity for learning new technologies such as PHP or .NET.

Project Scope

This project created the online application for the PAIR Test. The application implemented the current manual process of the PAIR Test functionality with additional functionality required for user, test and site administration. The application utilized Java technologies and a MySQL database. The application automated the entire testing process. The requirements implemented for this project are documented in the Chapter 3 Requirements section.

The product includes moving the existing www.pairtest.com from its current hosting company (Register.com) to a new hosting company. There was no interruption or loss of service to existing counselors using the site. The email provider was also migrated to the new hosting provider. This project enhanced and assured backward compatibility to the current PayPal online payment processing interface.

This project is defined as being successful if all of the above mentioned items are completed according to the plan. Success included implementing the functional requirements and delivering a quality testing process that can be used as a replacement to the existing testing process.

Summary

This project provided the ability to allow the PAIR Test continued growth to its business owners and clients. This project enabled clients and counselors to be geographically diverse and flexible. The final product provided a significantly enhanced technical platform on which to

support the growing needs of the business. The project gave the author the opportunity to increase his skills in the area of open source technology experience, which is becoming a key asset in today's technology sector. The author also gained significant insight into the deployment and operations of a custom built business application. This experience was invaluable to the author both professionally and personally.

Chapter 2 - Review of Literature / Research

Literature and Research Overview

Commercial Product Research

After researching through several products offered for testing software packages, there appears to be no solution currently built that can be used as a replacement to building a proprietary software solution. The author reviewed three products that were potential solutions for the PAIR Test vs. a custom built application. The products reviewed were:

1. CQuest Assessment Software by Cogent Computing Corporation (see www.cquestsoftware.com)
2. Perception Online Assessment Software by Pearson Assessments Inc. (http://www.pearsonncs.com/home.htm)
3. Perception by QuestionMark Inc. (http://www.questionmark.com/us/home.htm)

The author reviewed and researched the above products that provided online testing capabilities, and compared the three based on the following criteria:

1. Does the product allow unlimited number of testers
2. What are the costs per tester
3. Do they support different user types (testers and counselors)
4. Does the product support grouping questions into categories
5. Does the product support generation the relationship profile PDF
6. Does the product support charging the tester a fee for taking the test
7. Does the product support users to create their own account

The following table documents the finding for the comparison and review of the products listed above. The numbers on the top represent the criteria item from the list above.

	Criteria						
Product Name	1	2	3	4	5	6	7
CQuest	No	No	No	No	No	No	Yes
Pearson	N/A	No	No	No	No	No	Yes
QuestionMark	Yes	No	No	No	No	No	Yes

After completing the review and comparison above, none of the packages provides the major pieces required. These products have the capability to deliver a set of test questions to users over the internet.

These software packages are managed by an administrator and a fixed number of clients are allowed to login. The test can handle true and false type questions and stores the data in the database for reporting. The product costs were much higher than developing a custom application. Also, these commercially available systems did not meet many of the requirements of the system. Users could not self register and had to be created by the administrator. They did not support the idea of counselors and testers, only testers and an administrator. Lastly, there was no way to create the relationship profile from the available reports. All the reports provided were predefined and did not allow this type of report generation. For these previous reasons, using one of these commercially available products was ruled out as a possible solution for this project.

System Design and Open Source Research

Based on the requirements and the author's experience with Java, the author and the business owner agreed that the system could be built using Java and Java compatible technologies. The owner also stated that limited capital investment existed for this project. Therefore, a major requirement of the project was to minimize costs. The author began researching and educating the business owner on Java technologies and the benefits of Open Source projects. Normally, Java is not the best technology to implement a website for someone that is non-technical. The Java programming language is a very complex programming model that requires a very specialized skill set. There is significant cost and time

investment in learning how to develop and support applications written in the Java language. This project did not spend time researching other technologies that could have been used instead of Java. However, Java technologies offer the richest component libraries and open source framework components than any other language. To find out more about available open source components visit www.java-source.net. There are many design patterns and open source framework code components which allow developers more time to work on implementing the business functionality than any other language on the market today.

First, the author defined and researched architectural patterns related to Java based web applications. The author utilized the white papers from www.TechRepublic.com and Sun Microsystems Blue Prints site (http://Java.sun.com/reference/blueprints/index.html) to begin developing the high level system design and architecture. The article "*Designing Enterprise Applications with the J2EE™ Platform"* available on www.Java.sun.com website is a very good resource to help begin the overall design. The author chose to use the Model-View-Controller (MVC) pattern. This pattern was documented in the white paper "*Make your Applications Strut*" written by Jamie Scheinblum available on TechRepublic site. As quoted by Jamie, "The MVC design pattern provides a foundation for building extendable, reusable code. Part of the attraction of the MVC is that it forces you to abstract your code and think of your project in parts-presentation, work horse code, and the

orchestration of the two." The model aspect of the pattern is the business logic that handles the processing of data based on a request or input from the user via the user interface of the system. The view is the user interface or display layer of the system. Finally, the controller is the piece that handles routing and orchestration between the view and the model. Both the articles from Sun Microsystems and Jamie Scheinblum recommend using Struts for the framework to implement the MVC pattern. Struts is a Java-based framework that implements the MVC pattern to build web applications that promote reusable object oriented code to separate the business logic from the display or view. The next part of the MVC architecture pattern is the Model.

The Model is made up of business logic and the data required by the system to support functions of the application. The business logic was written in Java running in the Tomcat application server. Tomcat is an application server where the Java code is executed. The business logic layer must have a permanent data storage mechanism for persisting important data used by the application. The database can be a traditional Relational Database Management System (RDBMS) that uses SQL query language to manage the data stored in the database. Another option is to use an Object/Relational Database Management System (ORDBMS). The first option, RDBMS, requires that the business logic layer include an object-to-relational mapping tool to persist the Java objects to the RDBMS. This is because the RDBMS uses SQL language query to

retrieve the data as text or raw data. However, using an ORDBMS eliminates the need for this as it uses a user-defined object-to-database mapping catalogue. The ORDBMS uses this catalogue to process and return objects to the Java tier rather than raw data as with SQL RDBMS's. The author chose to use a traditional RDBMS for this project. The author chose MySQL RDBMS for this project as he had experience with this RDBMS and would save time and reduce risk to the project. MySQL is an open source database that is used and supported by many hosting companies. As stated above, using an RDBMS requires an object-to-relational persistence model. This can be obtained by using many technical approaches. These approaches and how it was implemented are documented in the Chapter 3 Design Section below.

Project Specific Research

View, Controller and Development Environment Overview

The author decided to move forward with implementing the Struts framework for the Controller and View components of the design. The Model was implemented using Java, Hibernate and MySQL database server. The author had never developed using the Struts application framework before this project. However, the author had previous experience with a proprietary developed framework controller. This helped with the understanding of the Struts framework, but there was still a lot of required learning of the Struts framework. The author started by

using *Struts in Action, Building Web Applications with the Leading Java Framework, by Ted Husted, CedricDumoulin, George Franciscus, David Winterfeldt, Mannig press.* This was a very useful resource in the understanding of the Struts framework and how to design and implement the framework. This book talks about how to get the framework installed and running, and what is needed to begin developing the web application. The Struts framework uses a Java servlet for the controller and JavaServer Pages to implement the view elements. The servlet is called the ActionServlet and reads an XML file called the struts-config.xml file. The ActionServlet processes requests from http that have a named action. The action names are mapped to Java classes called action classes. The mapping is in the struts-config.xml file. The mapping does more than just map the requested action to a Java class. It also defines where to go next in the flow. The mapping uses a string-like “success” or “failure” or some custom string that defines the next action to take - either calling another action or returning to the view for display of the data. There is also another element that the mapping definition does as well. It defines another Java class called the FormBean or Form. The Form is a Java object that is created by the ActionServlet when processing a request. The servlet populates page data that was entered in the web page into this form, and then provides a call to a validate method that then validates the data for required fields, length, credit card numbers, etc.. These things are all mentioned because these are all the elements of the

framework that used to be hand coded for each developer for each project or company. This is a huge benefit to developers to not have to spend time coding frameworks as part of a project. Thus, it leaves more time for developers to focus on the business functionality and developing the actual web application code. The next step of the development phase was to get an Integrated Development Environment (IDE) to create and maintain the Java code and Struts elements. The author researched many different IDE's and there were many good ones that did Java and Struts. The IDE's reviewed for this project were MyEclipse, IntelliJ® IDEA, and NetBeans. After researching these, the author chose a semi-open source product called MyEclipse. This is an extension to the completely free Eclipse product. This is based on the free open source Java IDE called Eclipse. MyEclipse is a small company that created a value added product on top of the Eclipse IDE. The MyEclipse product added wizards, tools, and plug-ins that makes creating Struts and other Java based open source frameworks simple. They provide a series of training videos and documents to help guide the developer through the using of the tools and wizards. MyEclipse costs $52 for the professional version and $30 for the standard. The author chose this IDE because the amount of features available were much more comprehensive than the others for less cost. Also, the author had used the base/free open source Eclipse product for about one and half years previously. IntelliJ IDEA was $299 for the professional version and $59 to get the standard version. The author did

not choose this IDE because this was too much for the author to spend on an IDE, and the author had no experience with this product. Lastly, the author did not choose NetBeans because the plug-ins for other open source components were not as comprehensive MyEclipse. NetBeans also did not provide wizards and plug-ins for Hibernate development.

Model and Database Overview

The Java platform provides several mechanisms for persisting object data to a database. The first and oldest technology used is Java API for Database Connectivity (JDBC). This is the most work for a developer, as this technology does not assist the developer in bridging the differences between relational models vs. object models. This approach requires that developers convert Java objects to tabular data to be inserted into the relational database. The developer also must create Structured Query Language (SQL) queries to get the data to and from the RDBMS. This requires the developer to understand the database structure and how to develop the SQL code. The second technology that is somewhat newer (2002) to Java platform is Java Data Object's (JDO). "JDO is a specification to enable transparent persistence of Java objects" as written in the article "*An Introduction To Java Data Objects*" by Jeff Brown, Senior Software Engineer, Object Computing, Inc. (OCI) (http://www.ociweb.com/jnb/jnbJun2002.html). This article gives a quick overview and comparison of JDBC and JDO. The author found this article to be a quick helpful read and introduction to JDO. It also lists a few

commercial JDO implementations and links to the company websites. JDO uses transparent object persistence, which allows developers to code to Java objects and not worry about learning the syntax of SQL or other database specific query languages. The underlying code of JDO takes care of working with the database for processing query and management functions. JDO has not been widely adopted by the Java community and developers as of yet, and the JDO spec has not defined the mapping configuration, so there are different mapping configurations for each implementation. These issues have caused developers to move away from JDBC and JDO toward a newer technology in the market place called Hibernate, which is an open source object-to-relational mapping tool. Jeff Hansen wrote the following quote in his article "*Simplify Java Object Persistence with Hibernate*" (http://www.devx.com/Java/Article/22652/0/page/1), which defines the basic reason why Hibernate exists and is used by developers.

"The architectural differences between Java object hierarchies and relational database tables make the task of persisting Java object data to and from relational databases quite daunting for developers. The "impedance mismatch" between relational tables and Java object hierarchies has led to the development of several different object-persistence technologies attempting to close the gap between the relational world and the object-oriented world. The Hibernate framework defines an object/relational mapping mechanism and query language that makes storage and retrieval of Java objects to and from a data store a relatively simple proposition."

Hibernate allows developers to write very little, if any, SQL code or proprietary interfaces or classes for database access coding. Hibernate uses an XML file to map the database connection URL and the object mapping to the database schema. It then uses Java libraries and the Java reflection package to implement the necessary Java coding and interfaces at runtime, which is all transparent to the developer. The author has chosen Hibernate for all database management and object-to-relational mapping. Also, MyEclipse provides plug-ins to generate the Hibernate XML file and related objects from an existing database schema.

Other Technical Research / Overview

This project required the ability to create a Portable Document Format (PDF) file on the fly. This was driven by the fact that a counselor currently uses a manual PDF document they print and fill in to display and use as a tool for consultation with their clients. This was an area that the author had no experience. The author began by using the Google search engine for “Java api pdf generator”. There were several results returned - DynamicPDF™, IText, and Big Faceless Java PDF Library. After reviewing the associated websites, there was a clear winner. This was iText, found at (http://www.lowagie.com/iText/). The other two possible solutions were too expensive for this project and the free iText could be used to implement all the needs of the project. The web site has many links to documents and other related information, but what the author

found to be the most helpful was the page for examples (http://itextdocs.lowagie.com/tutorial/). This page provided almost everything the author needed to create the PAIR2 Relationship Profile PDF shown in the appendices below in Chapter 6.

The user-interface was to be re-architected to support the needs of multiple user types. The PAIR Test website prior to this project only had counselors using it to order paper materials for the PAIR Test. This project changed the site to also serve clients using the site. This required the separation or hiding of certain data from the clients, such as the cost of the tests and how the counselors scored and used the tests. This was specific information that a tester should not see. Therefore, this required the website to be re-designed to have data easily exposed and accessible for the user type. The author decided to use a few new technologies that made redesigning the user- interface much easier. First, prior to this project, the content for the site was based on hard coded images which did not render well in different browsers. The second issue was the fact that there was a lot of redundant HTML coding and styles used in building the interface layout. This did not allow for changes and dynamic content display based on user type. The first step was to convert all the HTML pages to JSP pages and not use images for headers. This meant using text for hyperlinks that would display better and look more professional. The next step was to decompose JSP's to separate the look and feel from the functional elements of the page. The author then implemented Tiles

and Cascading Stylesheets (CSS) technologies into JSP's. Tiles is a templating framework that is an open source sub-project from Apache Struts. It can be used to create reusable view components that control layout and components for a standard look and feel across your entire website. For more information on the features and usage of Tiles, see the Tiles homepage Apache web site (http://struts.apache.org/struts-action/struts-tiles/index.html). CSS acts as a partner to your HTML code, taking care of all the layout, fonts, colors and overall look of your site. Stylesheets allow you to separate the style and layout of your HTML files from their informational content. CSS allows one place to change the look and feel, less coding in the HTML tags, and less maintenance when the look and feel changes. CSS can decrease page loads by as much as 50% in some cases. CSS also helps handle display issues across multiple browsers, which allows better user experience and accessibility for users using the site from a PDA or handicapped device. For additional information on CSS and the benefits of CSS, see these resources: "*Benefits of Cascading Stylesheets*", by Alan Richmond (http://www.wdvl.com/Authoring/Style/Sheets/Benefits.html) and "*Introduction to CSS*" by Ross Shannon (http://www.yourhtmlsource.com/stylesheets/introduction.html).

Project Topic Knowns and Unknowns

We know that this is a project with a timeline and delivery for the author and once the goals and objectives of this project have been

completed, the application maintenance, support and ownership will be done by the customer (Dr. Gene Mastin). The PAIR Test organization and Dr. Gene Mastin do not have the technical expertise to maintain and support a self-hosted application. Therefore, this project will be hosted by WebAppCabaret. The cost of hosting is much less than the cost of hiring or learning the required skills for Dr. Mastin or the PAIR Test organization to be able to manage the applications production environment and website operations. Dr. Mastin will be trained by the author in how to use the software and database management tool provided by the hosting company to manage and maintain the website, application, and email accounts. This project will be turned over to the customer after the beta period has completed and all issues resolved. The source code will be delivered along with any related how-to guides and database creation scripts packaged in a zip file. The customer can then give the zip file to another student if he chooses to continue using academic resources for continued development of the PAIR Test application. The author has offered to continue to provide consulting to the customer for a nominal fee, however at this point no commitments have been made or discussed.

This project is not inventing a new database or programming language. This has already been done and is not the best use of the time for this project. The project used many open source components, which had several books and documents containing how to use and build applications using these components. This is already something that has

been available for years in the open source community. This project made use of those resources to help in the successful delivery of implementing and integrating these components to deliver a success business application for the PAIR Test.

What is not known about this project is the business and its proprietary database, business logic for the test procedures, scoring algorithm and the customer/user base that uses the system. This required a proprietary database and application that implemented these rules.

The Contribution this Project will Make to the Field

This project did not develop a new programming language or operating system that will have an impact on the industry such as Java or Linux had on the field. The impact will be on people and students at Regis and universities faced with the same challenge. They could use this document to define a blue print and/or recipe for building an application using these technologies and redoing much of the research performed for this project. They can also start implementing the details of the application and have known issues to consider, rather than start from scratch and not make the same mistakes made early on in this project.

Chapter 3 – Methodology

This application enhanced the PAIR Test by making it available to couples and counselors saving the need for printing, shipping, and manually processing the tests. The new online system has already saved

significant time for all parties involved in the PAIR Test. The system allows counselors and couples to not have to physically meet, and allows flexibly in geographic locations for couples choosing a counselor.

Research Methods

Research for this project used client interviews to help define and educate the author on the system requirements and what the business is about. The project used a variety of resources for technical consulting. These included open source project web sites, internet search engines, and formal publication at the local library. The project also used the internet to research hosting companies that could provide the services required to deploy and run this application.

Project Approach/Methodology (How)

This project used a mix of Agile/XP and traditional Waterfall SDLC methodologies and techniques. This allowed the mitigation of risk involved in the introduction of new technologies to the author. Once the technology and hosting company were selected, a small prototype was created and deployed to the hosting platform to allow discovery of potential issues and impact to the project. The results of the prototype determined whether the selected technologies can be utilized for this project, or if a different hosting company and/or technology should be used instead.

This project had the following phases:

1. Requirements and scope definition of functionality
    1.1. Inputs
        1.1.1. Existing hardcopy tests booklets
        1.1.2. Existing relationship profiles
        1.1.3. Emails documenting functional processing
2. Analysis and technical approach – research of required technologies to meet the functional specs
3. Determine the hosting options and decide hosting company
    3.1. Compare Register.com (current PairTest.com provider) to other hosting companies to see if they have the needed support and products to support the project technology
4. Prototyping and hosting feasibility analysis
    4.1. Apply findings to project
5. Design, construction and testing
6. Develop user acceptance test plans and test cases
7. Alpha deployment for testing on hosted hardware
8. Client acceptance testing
9. Beta test – small group testers
10. Final deployment

Metrics/Measurements – the project was ready for beta test once the system accurately implemented or provided the functional item listed in

the Requirements section below and passed the Client Acceptance Testing phase.

Requirements

Requirement ID	Functional Element	In/Out Scope
1.	The system shall provide for users to self-register and take the test.	I
2.	Each user will require a counselor ID to create their individual user account.	I
3.	Each user will be required to pay a fee based for each new account they create.	I (configured to off or 0 in the DB for launch)
4.	The user account and testing fee will be processed via PayPal.	I
5.	If a user cancels the PayPal transaction then the user will not be allowed to take the test.	I
6.	The fee for the test will be configured in the database and the administrator will have the ability to change this value. If there is a 0(zero) amount for the fee the system will not try to process a fee.	I
7.	The system will allow for the testing fee to be turned on or off. The administrator can set this value to ON or OFF by changing the fee amount in the database. A fee of greater than 0 (zero) will indicate to process a fee and the amount, a fee equal to 0 (zero) will indicate not to process a fee.	I
8.	Users are not allowed to take the test more than once.	I
9.	Users are not allowed to “go-back” on a question.	I

	Once they have answered a question the result is recorded and the next question is presented.	
10.	Users are not allowed to start and stop the test. The test must be completed in one sitting.	I
11.	User accounts will be stored in a database.	I
12.	User accounts will have the following attributes:	I
a)	First and Last Names	I
b)	Date of Birth	I
c)	Counselor ID	I
d)	Unique User Name	I
e)	Email Address	I
f)	Password	I
g)	(8 characters, 1 must be a uppercase, and one must be a number)	I
h)	Gender	I
i)	Marital Status – Single, Married, Divorced, Widowed	I
j)	Payment Received – True (t) or False (f)	I
13.	The system shall provide the ability to change/update all user account elements above except – Unique User Name. An admin may delete a user and re-add that user, but doing so also deletes any test results previously stored for that user account.	I
14.	The system shall provide the ability to display all users, counselors, and admin accounts.	I
15.	The system shall provide the ability to create new Legacy Counselor accounts.	I
16.	The system shall provide the ability to create new Pair2 Online Counselor accounts.	I
17.	The system shall provide the ability to create new	I

	Admin accounts.	
18.	The super administrator or admin users will create all Counselor/Admin accounts.	I
19.	Counselor/Admin accounts will have the following elements and also stored in the database:	I
a.	Counselor/Admin User Name/ID	I
b.	Password	I
c.	(8 characters, 1 must be a uppercase, and one must be a number)	I
d.	First and Last Name	I
e.	User Type	I
f.	Legacy – only allows ordering hard copies of test materials – no online participation	I
g.	Pair 2 Counselor – only allows access to retrieve results for online relationship profiles for users associated to this counselor	I
h.	Admin user – has ability to add/delete/modify Users, Counselors, Questions, Test Results for users.	I
20.	Counselors wanting both user types will need 2 accounts.	I
21.	The system shall provide the ability to add/modify/delete questions from the database.	I
22.	Questions will have the following elements:	I
a.	Question ID	I
b.	Question Category	I
c.	Question content (the actual question)	I
d.	Active flag (is this question visible to the tester)	I
23.	The system shall provide a list of questions to admin user.	I

24.	The system shall not allow adding new categories.	I
25.	The system shall not physically delete any question; only mark it as “inactive”. This is to allow changes and removal of questions but test results still need the old question for scoring.	I
26.	The system will not track or validate the number of questions per category/scale.	I
27.	The system will present all “active” questions to the user. Deleted or “inactive” questions will not be shown to the user.	I
28.	Modify questions only allows the question content to be changed.	I
29.	The scoring will happen by only counting the number of “true” responses for a category. Questions answered as false are stored in the results database but not factored into the scoring.	I
30.	The profile will not plot the points on the graph. Instead it will create a PDF that has all the content in Appendix A: PAIR2 Relationship Profile excluding the graphed points in the center. The counselor will print the profile pdf and manually plot the graph points. See figure 1.	Partial

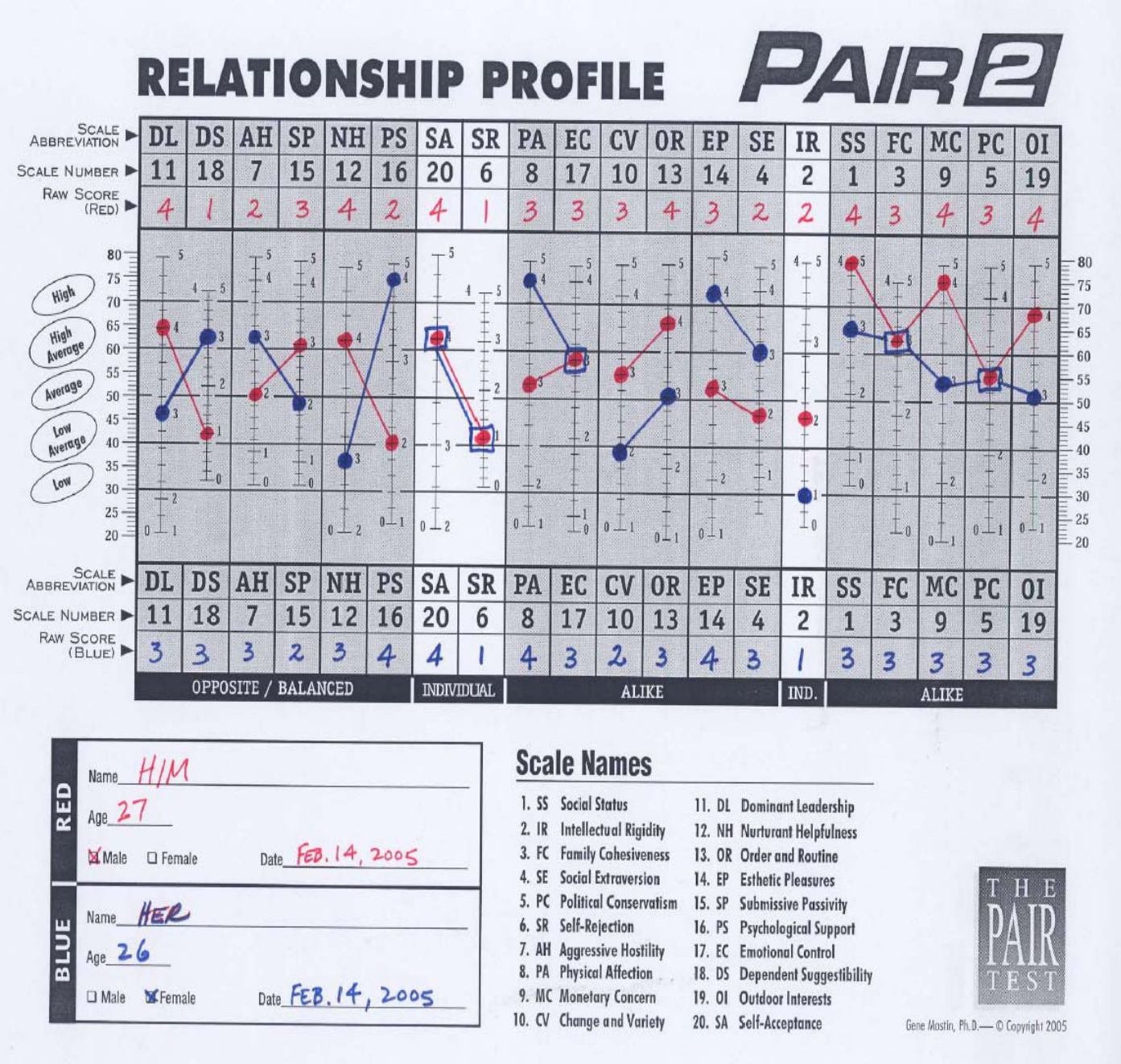

Figure 1. *Sample PAIR Test relationship profile form. Prior to this project the counselor would take this and manually complete all the areas in blue and red. This project built this form as a printable PDF and automatically completed all the blue and red data using data from the database. iText was used to create the PDF document.*

Analysis

After completing the requirements definition phase, it was time to start the analysis. The analysis phase deliverables were to define the use cases, high level system architecture and persistent data elements. Also, this phase analyzed the existing web site pages and code to see what could be reused in the system. Secondly, the author began the technical analysis and planning required to develop the prototype using the selected open source technologies. The requirements were used to begin defining the use cases and persistent data elements required by the database.

The following describes the system architecture:

A user sends an HTTP request from a browser to the Struts action servlet. The action servlet reads a struts-config.xml and finds the associated mapping for the request. The servlet then calls the business layer to process the request. The business layer may call the database tier if data operations are needed for this request, otherwise the business layer tier processes the request and returns back the next action for the action servlet to execute. These actions may call another business tier class or a JSP to render the html back to the client browser. Eventually, the final action will result in a JSP call that returns back to the client browser. This flow is depicted in figure 2 below.

The diagramming needs of the project were completed by using the MyEclipse UML diagramming capabilities. The MyEclipse IDE was very easy to use and does full object-oriented system modeling and design.

The IDE allows for both generation of Java code from a model or reverse engineering Java code into a model.

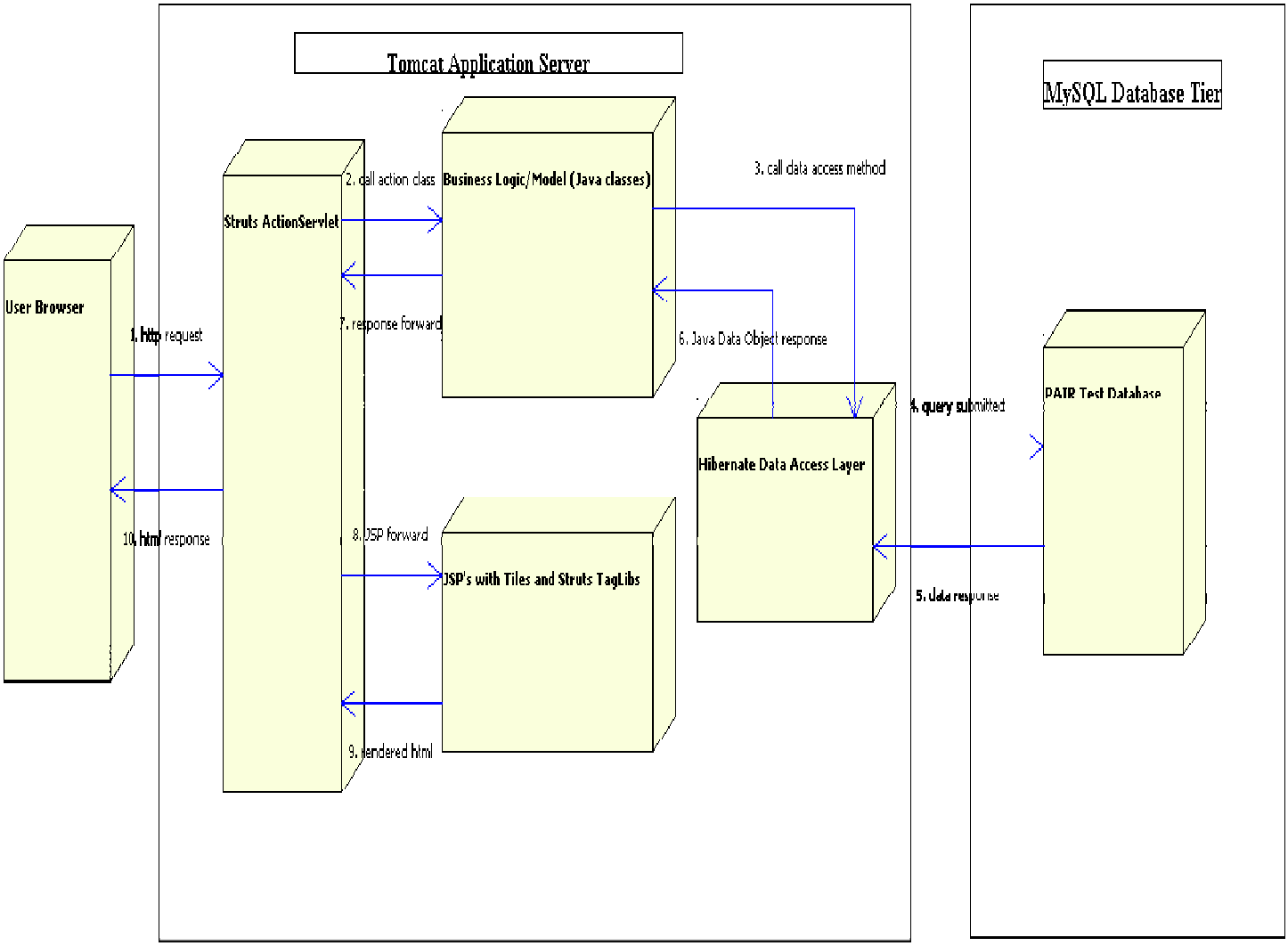

Figure 2. *PAIR Test online system overview/component diagram.*

Use Case Models

The Use Cases were developed using the business requirements. The business requirements defined what the user interactions with the system would be and they defined those down to a level to begin

developing the software solution. Below are the Use Case models for the system. The requirements defined four actors or user role types that would be accessing the system. The diagrams in Appendix A depict the role types and their associated Use Cases. The actual Use Case documents were too big to be included in this document, however the template and example is attached in Appendix B.

Data Requirements and Modeling

The persistent data elements identified from the requirements were related to clients/users, questions, test results, and configuration data. The data model needed to build relationships between client/users, counselors, and the test results. The validation rules are also that user name is an email address and must be unique across all users. A client/user is related to the counselor via the counselor id element of the user entity being the actual counselor's user ID. There was not a counselor's table and a user table. There will be one table for clients. The client table will contain a user type to make the distinction between counselors and test clients. This was done because the attributes are very similar and would save on the coding and database space on the server. The only data that was implemented for the configuration entity was the fee for testing. This was implemented in the database so that the business owner could easily change the amount. It also allows not redeploying the software to change the amount. This could have been done using a properties file vs. a database, but that would not allow

changes without rebuilding and deploying the application, which is beyond the technical expertise of the business owner.

User Interface

The user interface or web pages of the old site were not reusable for the new user interface. The html was very poorly written and would not convert into JSP's cleanly. Therefore, every existing page was converted into a new JSP. This was also required because a user's role controls what pages to hide and show. There would be one header applied to all pages that controlled what pages a logged in user or non-logged in user could see. The layout of the new site was completely rewritten to have a more professional look and feel. The old site was also not capable of hiding and showing data based on user type as the pages were static HTML.

Design

After completing the analysis model, it was time to begin the design phase. The design phase for this project was not the traditional model for design. This phase started off by developing the prototype. The prototype was scoped to contain only the necessary design and code elements needed to build the functionality for the login process. Once the prototype was completed, the traditional waterfall design approach was used to design the remaining functionality. When completed, this phase

documented the user interface definition and layout, database design, Java object models for the business layer and data access services.

User Interface

The user interface design will include the JSP's, Tiles and the Stylesheets. The article "*UI design with Tiles and Struts"* by Prakash Malani (http://www.Javaworld.com/Javaworld/jw-01-2002/jw-0104-tilestrut.html) documents several techniques for layout view components. The view components are very modular and are difficult to diagram. Therefore, to illustrate the design, a diagram was not used. Instead, a brief description of the components was given of each, and the complete login code will be shown. "Tiles" is a layout engine that works with the Struts framework. "Tiles" is used to assemble components at run time via a definition file. In this design, a page is made up of a pagedef, a layout, and several modules. A pagedef is the definition of a page. It uses Tiles insert to define the view components and layout component. The layout is used to arrange the view components. The modules are small reusable view components or blocks of html elements. In the login example, the login.jsp is the Tiles definition, the layout.jsp is the layout manager, and the header,jsp, loginModule,jsp, and footer.jsp are all view components. The source is attached in Appendix C. According to Prakash Malani, this design is not the recommended approach for a Tiles and Struts application, but does offer good code reuse, and loose coupling between view components. The recommended approach was not used for this

project to eliminate another layer of complexity that did not change the design or functionality. Here is a screen shot of the new home page for www.pairtest.com. See figure 7 for the PAIR Test home page.

Figure 7. PAIR Test home page image.

Business Logic / Data Access Layers

The second part of the design is the business and data access layers. The components of the model tier are Java classes that implement the business functionality. The result of this part of the design is a class diagram. The class diagram describes the associations, data and behavior of the Java classes that implement a set of functionality for the system. This diagram shows the elements used to implement the Login function. The action servlet is not included in this diagram because it is

part of the Struts framework and not part of the model tier. The classes shown here are responsible for accepting the user input and validating it against the database for accuracy. This is the design pattern used for all functions of the model tier. This design pattern is used to separate business logic from data access logic. The separation occurred because the database interactions are done behind a service that takes business domain objects and returns business domain objects. The data persistence could be changed without impact to the business layer. The class diagram is depicted below in Figure 8.

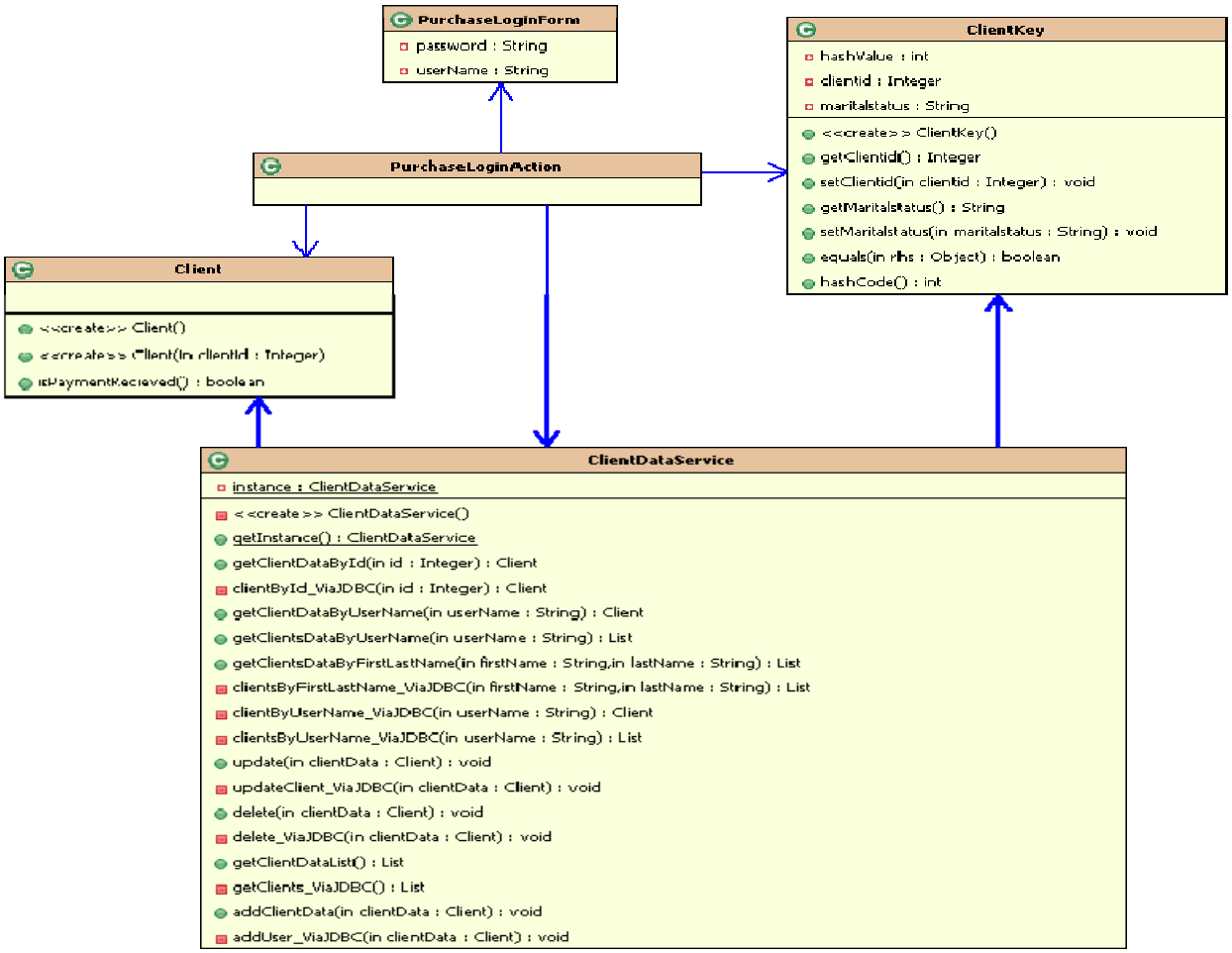

Figure 8. Login Class diagram.

Database Design

The goal of the database design was to provide an easy to understand database schema that provided reporting on the test result data, and which allowed the database to be extended easily. The database was also relational so the relationships were implemented using foreign keys from the client and question tables to link the results of a client to the question database. Figure 9 depicts the database tables designed for the application database.

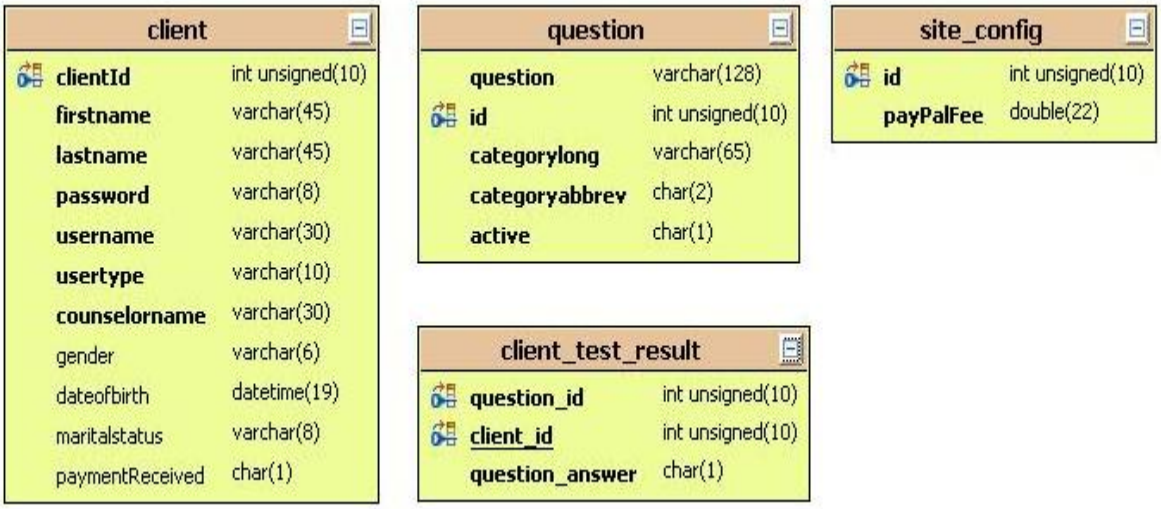

Figure 9. Database design and tables.

Life Cycle

The SDLC used for this project was a mixture of both Extreme Programming (XP) and Traditional Waterfall. As documented on Extreme Programming Organization website (http://www.extremeprogramming.org/what.html), “Extreme Programming (XP) is actually a deliberate and disciplined approach to software

development". XP enables developers to keep designs simple, test software early, and use the results as feedback to detect issues and changes early in the implementation cycle rather than later where changes are more costly. The author chose to use XP in the early stages of the project to get a prototype competed to ensure that he understood the open source frameworks. These open source frameworks were new to the author and this was a way to learn them and gain understanding and familiarity with them before starting the real design phases. Once the prototype was completed, the project moved into the traditional waterfall method where the author developed the requirements and scope of what the system would do as documented in the Requirements Section in Chapter 3.

Procedures

The procedures for this project were to get the prototype built, define the scope, and develop the application software using XP and Waterfall methods. The prototype was developed using the initial conversations and emails documenting system functionality by the customer. The customer and the author agreed upon the scope and functionality. The author's advisor was also involved in reviewing and approving the plan and giving advice on areas such as scope control, mentorship on focusing on the goals, and handling of setbacks and issues. The initial testing/QA level testing for this project was completed

by the author and the customer as well. The beta level testing is currently being performed by a group of university students.

Deliverables

The formats used for producing the project deliverables were Microsoft Word documents for the project proposal, which included the project plan that was completed using Microsoft Project. The analysis phase provided a detailed software design document, system interaction diagram, deployment model, database model, and any other additional documentation required to accurately document how the system works.

Deliverables Review

All of the deliverables were met with the phases identified. The prototype deliverable took longer than expected. The delay was in getting the application deployed on the hosting company's server. The development of the prototype code was on time and went according to plan in the development environment.

Resource Requirements

The resources to complete this project were the author and the customer. The customer's role was to provide the domain knowledge and the functional requirements and approve or reject the user interface designs. The author's role was to be the functionality analyst, functional documenter, designer, developer, and tester. Tech support was needed

from the hosting company and this proved to be a resource that was not initially anticipated, but assisted in the success of this project. For the design, development and deployment of the project, the internet had valuable documents that helped solve issues quickly. Also, some other technical books relating to Struts, Hibernate, Tomcat, MySQL and CSS were used for this project.

Outcomes

The expected outcome of this project was the successful delivery and deployment of the PAIR Test application. The application helped reduce costs and provided additional benefits to the PAIR Test users and owners. The benefits are expected to be a significant saving in time for completing, administering, and distributing the test materials and process. This project reduced costs for the PAIR Test owner. The cost saving is based on the fact that there is no longer a need to ship paper materials, and the project will deliver a more scalable process to handle the significant growth for the number of counselors and users. This project also produced a sample technology solution that future students could use for a template and guide for implementing future projects at Regis University.

Summary

Using an Agile/XP methodology that consisted of researching new technologies, implementing a prototype, and moving into a traditional

Waterfall approach worked great for the completion of this project. It allowed the project to define and minimize the risk of creating code that would need to be re-coded or replaced. The project consisted of many new areas to the author. Following a traditional method would not have allowed the author to gain enough understanding in these areas to develop in a practical sense. Having been able to get hands-on with new technologies and hosting companies, the author was able to expose issues within these areas that would not have been understood until the end of the project, where redesign is more costly and would have had more impact on the project to adapt change. The project was able to complete a better design and implementation from the learnings of the prototype. The deployment of the hosting company was also a learning experience during the prototype phase that proved to be invaluable, as this allowed the author to understand what aspects of a hosting company would be important to the successful deployment of the application. Overall, the methodology for the project was a success as a combination of Agile/XP and Waterfall. The choice of Hibernate and Struts were great even though there were issues. These issues were mainly related to the hosting company offering. Hibernate and Struts decrease the amount of code needed to be developed for a Java web application and the author will plan on using these as a part of the next project.

Chapter 4 - Project History

Beginning the Project

This project was initiated by Associate Professor Dr. Gene Mastin from Liberty University. Dr. Mastin is in the Center for Counseling and Family Studies Department. He owns and administers the PAIR Test, a psychological/relational inventory. The PAIR Test assesses compatibility issues for premarital and marital therapy. Professional counselors, psychologists, pastoral counselors and military chaplains use the test as a tool to aide in understanding personality traits of couples. Dr. Mastin contacted Trisha Litz at Regis University, who indicated that there may be students looking for graduate projects. The author then contacted Trisha Litz for possible projects in December 2004, which is when the project started. Dr. Mastin's request to Trish Litz was relatively simple. He indicated that he was interested in exploring the possibility of placing the test on his web page (www.pairtest.com). The web site would provide the ability to complete the test in an automated application, automatically calculate a score, and complete a couple profile. The counselors could also go to the web site to retrieve the relationship profile results and print them out. At present, all scoring and profiling is completed by hand.

The following quote from Dr. Mastin explains his original vision - "to enable a couple to actually take the test online, simply by clicking their mouse on True or False on the screen, and then the program takes them

automatically to the next question, with no returns or going back over the test. They would simply log out of the site, and the program would calculate the results and produce a Profile Form. The couple's counselors can then go on the web site and with appropriate passwords, call up the Profile and print it. In order to do that, the counselor - or the client - would have paid the fee (by credit card) for the test. Payment would allow all of the above to occur. I have not yet decided on an amount for the testing fee, but that can come later." Thereafter, the author began planning for the project in January 05, however, the only progress made between January 05 and October 05 was the author and Dr. Mastin shared some emails about the functionality of the current test. The author also had begun researching some potential technical direction for the project. See the Research section above. The project began in November with the project being approved by Trish Litz, the Regis graduate professional project advisor.

Project Management

The author was the developer and project manager. The author did not have experience in setting schedules, managing risks, and understanding scoping and level of efforts planning associated with a project of this nature. Therefore, the initial schedule was not very feasible and the author soon realized that issues had significantly invalidated the project milestones. The prototype phase provided for technical mitigation and then to move into a traditional waterfall project methodology. Due to

scope creep and delays due to learning new technologies, the project began to get out of control. The author approached the advisor to gain some insight into getting the project back on schedule and on track. The advisor suggested locking down the scope and then defining a plan to get there. The author then created a document containing all the functional elements of the system including the required persistence elements of the database. This was used to lock and control the scope of the project. Once this was completed, the author continued to work on the implementation of the prototype.

Significant Events/Milestones in the Project

This section describes the significant events and milestones for this project. The following events were significant milestones during the course of this project:

1. Researching and choosing framework components
2. Defining and developing the prototype
3. Deploying the prototype application to the servers
4. Analyzing and designing the remaining functionality
5. Completing the development
6. User acceptance, layout and usability testing
7. Beta testing
8. Final Deployment

The first milestone for the project was to complete the research and choose framework components by determining the framework and technologies that would be used for the web application tier and the database server. Next, was to install all the components needed for the application (i.e. Struts, MySQL, Hibernate, and MyEclipse). Once this was completed, the defining and developing the prototype phase could begin.

The defining and developing the prototype phase consisted of building enough software so that the critical path, proof of concept, and technical feasibility could be exercised. This was completed when the questions and answers were successfully read and stored in the database from the web application. This was enough software to prove the assumptions and outcomes of the researching and choosing framework components phase. Once the software for the prototype was completed on the author's computer, the next phase was to deploy the prototype application to the servers and get the prototype executed on the hosting company's server.

Deploying the prototype application to the servers included setting up the hosting account, learning the hosting company's tools for deploying the software, and developing the database instance on the hosting company's database server. This milestone was completed when the prototype was executing properly according to the specifications. This milestone took much more time than originally anticipated (see Section entitled What Went Wrong below for details). Once this milestone was

completed, the analyzing and designing the remaining functionality of the web application was completed. Completing the development was the final phase before user acceptance testing could begin. This phase was completed according to the original timeline due to the fact that all the issues had been worked out in the deploying the prototype application to the servers phase. The next milestone, user acceptance, layout and usability testing, was defined as the customer would perform basic usability and functionality testing. This phase would define and document issues with layout and usability, and prepare the application for the beta testing phase. The beta testing phase is currently in progress at the time this paper is being written. Finally, the last phase, final deployment, is planned to be completed once any issues are documented and resolved by the author.

Changes to the Project Plan

The project plan from a high level did not change much. However, that does not mean that the project was done according to the timeline. The plan changed to allow more time be spent on the selecting of the hosting company because there were many issues with the first selected hosting company (as documented in the What Went Wrong section below). This caused the project plan to change the dates for completion of the prototype phase and all other phases that followed.

Also, the original project plan changed to include design and construction time for the additional scope for administrative capabilities required that were not originally planned for this project.

Evaluation of Whether or Not the Project Met Project Goals

The customer was very pleased with the new user interface and felt the usability was simple and easy to follow. The site layout and navigation did not allow the user to get lost in what they were doing. This project had its ups and downs, but this was a very rewarding experience and the response from the customer was very reassuring that this project was a success. The personal goals experienced during the course of this project met all expectations by allowing the author to gain experience in project management and deployment aspects of the SDLC. This project offered a great learning experience for deploying a web application, selecting hosting providers and being more efficient prior to starting a project from a deployment perspective. Overall, the project met the goals of what was expected and delivered. The delivery dates and timelines were somewhat sacrificed along the way. This would not have been acceptable in a commercial project, but having an informal project in which it was stated up front that timelines would have to be flexible, enabled smooth working relations between the customer and the author.

What Went Right and What Went Wrong

What Went Right

After several months of delay, it took two weeks of reviewing different hosting companies' plans, capabilities and support models. A spreadsheet was built to track and compare the different hosting company capabilities and offerings. This was used to choose the company that offered the best match for the requirements needed to support the technologies used in developing the application. The prototype was successfully deployed and working within ten days of the selection and access to the new hosting company. This was a great step in the completion of the project, which allowed the project to get back on track. Finally, after the prototype was tested and the critical path completed and working, the project got back into the planned analysis and design phase.

The user interface redesign was a complete success. The user interface for the old website was targeted toward a user base of counselors and not end users. The site was mainly used to purchase materials via Paypal with one generic login id/password shared by all counselors. The new site had to be more user-friendly and allow users to easily register, take the test, and see other information related to the PAIR test. The new site had to hide content based on role type for administrators, counselors, and end users. The old user interface was very outdated and unprofessional in appearance. The implementation of Cascading Style Sheets (CSS) proved to be a great asset. This

technology provided the ability to enable changes to the look and feel and colors and fonts, in a simple and global manner. The author had no previous experience as a user interface designer and never used CSS before, but this was a very positive and rewarding experience. The customer had very positive feedback on the new site layout. After a few minor tweaks to gain the customer's acceptance, testing the site was made available for beta testing. Currently, there have been only two code issues. Training the customer is the only remaining effort at this time to complete the goals and requirements. The training is scheduled to take place once the beta testing is complete and the author has resolved any issues from the beta test. This is the point at which the customer is expected to own the operations of the site and the author has completed his commitment to this project.

Also, the new hosting company was great (WebAppCabaret.com). They were able to handle all my questions in a very timely manner. They provided the ability to upload changes and manual restart of the server, which allowed immediate testing of changes. This significantly improved the progress and turn around time for fixing issues and implementing the remaining functionality. They also enabled the ability to upload data files directly into the database, which also saved a significant amount of time in entering the 200 questions for the 20 different categories.

What Went Wrong

The author decided on Struts and Hibernate for the implementation and began working on implementing the application using these technologies. After few months of development, the author felt is was time to move the prototype/application code to the production server's environment. This demonstrated a serious flaw in the selection of hosting provider, but not until significant time (two months) was spent debugging issues with environment and application to get it working. The major issues existed because the author was not experienced with the deployment of Hibernate and Struts. Therefore, the selected hosting company was not appropriate for the deployment of these technologies.

Hibernate persistence framework was used to access the database. Hibernate uses a file to configure the database access methods, which allows less development from the Java developer. However, the hosting company and plan selected did not allow any files to be read from Java code running in a shared server environment. Therefore, the author was faced with either changing hosting companies or rewriting the code using a different technology. Switching hosting companies seemed to the author like a bad idea, as there may be different issues with a new hosting company. The author selected to rewrite the code using the JDBC technology. Thereafter, the author was able to get the login and the database connections working.

The second and final issue with the first hosting company was related to their security policies and the Struts framework. Struts code was used to build the user interface portion of the application. Struts is a very popular and easy to use Open Source MVC architecture framework. It provides significant productivity and time saving for developers allowing them to focus on creating code and not a custom framework. However, Struts uses a Java technology called Reflection. Reflection is a runtime operation that allows dynamic creating and manipulation of Java classes/objects to be discovered, modified and changed by developers at run time on the server. The hosting company did not enable the security needed to allow this to be used in a shared server deployment. This issue was too big to rewrite, so the author realized it was time to change hosting companies.

The selection of the first hosting company (GoDaddy) was a complete mistake. The core competencies of GoDaddy hosting company was not Java, and this caused serious issues in trying to debug the problems and why things were not working. Application support and deployment was also an issue. They had no experience in Java and could not help me get my application deployed. They also did not enable access to log files to see application errors. They also took several days or more to respond with feedback on questions. GoDaddy also did not allow restarting of the application servers by customers, therefore when a new deployment was uploaded, it would be the next day or longer before

the author could test the new or modified code. They also did not enable uploading of data files to the database, which caused manual entry of all questions into the database.

After the two month delay, it was evident that the selected hosting company was no longer going to be able to be used without significant work to rewrite and build new custom code. The decision was made to cancel this hosting plan and go with a new hosting company. Lessons learned from the first hosting company experience provided good information that was used to select the new hosting company. It is hard to say how this mistake could have been avoided because the author had to learn Struts and Hibernate prior to making a choice that worked the first time. However, spending more time delving into security policies and asking: do you support technology “x” or “y” i.e. Struts and Hibernate would have been the right questions to ask. The author assumed if the company supported Java, Tomcat and MySQL, then Struts and Hibernate would work.

Discussion of Project Variables and Their Impact

There were several variables that changed the course of this project. The scope was not locked down in a formal manner, so several times things had to be rewritten or modified after the functionality was reviewed with the customer. New technologies take time to learn and become proficient in completing tasks and this caused some normally trivial tasks to take longer. The third element was getting the customer to

understand what goes into building a website and database and to think about how and what would be needed to take ownership of the product. The customer had no previous experience in building software of this nature. This required the author to define the requirements and discuss them in great detail with the customer before the software could be built. The author's lack of available time to work on the project was another factor that contributed to the project taking longer than anticipated. Only being able to work on the project part-time resulted in several times when a few days or even a week went by where no progress was made. What seemed to work well was defining smaller tasks that could be completed in one work session.

Findings/Analysis Results

A significant finding/realization that occurred during this project was that choosing a particular technology based on popularity is not always the right way to go. Hibernate is an excellent object-to-relational mapping framework for database persistence, however many hosting companies wanted to charge more for applications using this technology, if they even supported it. Hibernate technology was possibly also not needed and caused more delay, and the same results were achieved using an older, yet more widely supported and understood technology.

Summary of Results

The project was very slow moving in the beginning and seemed to almost stop a few times. There were a few months the author spent working on testing out different ideas during the prototype. Looking back on it, the author saw that the methodology for this project worked. The author supports this by saying that the first three months produced painful results and not a lot of functionality, however the second half of the project was much more productive. The biggest step forward for this project was the day that the author got the prototype code completely working on the hosting company's production server. This started the completion of the rest of the real work that needed to be done, which was to create the design and implementation of the business application to meet the requirements and functionality of the system. This was a big step because there was no more worry about how to code certain pieces and whether or not it would work. The methodology used worked very well for this project. This method could be reused in the future on other projects and this experience and document would help be a guide. Of course, there are certain things that could be changed, but those will be discussed in Chapter 5.

Chapter 5 - Lessons Learned

What you Learned from the Project Experience

The author learned that project scope should always be the first thing that is identified and documented to a point where the entire breath of the project can be completed in the allotted time. Another valuable lesson was the developer realized that more time in the research and upfront planning will save significant time for the project. The author also learned that the deployment and maintenance of a project such as this should be planned into the project. Currently, the customer has no one lined up to make any future modifications. It would have been more effective to have documented an exit strategy very early on in the project. This would have communicated and set expectations in how the project would end. This was done, however somewhat informally, but the author expects that the customer will call with questions on how to perform a function or fix an issue. The author has set the expectation that support for code written incorrectly by the author would be fixed if it was not functioning properly according to the original specification, but no new enhancements would be included. The author has learned a significant amount about how to control and manage future projects. The author has learned what things need to be decided first and what the final product will look like so that appropriate planning can take place to prepare for the final production state, and how the ongoing maintenance will be handled.

These are all items that in the author's professional career as a software developer, he has not had exposure to. This project has given the author insight into being more proactive in his professional career and personal growth in these same areas. The author also feels that this project has opened the door to start doing some independent consulting in the future, which has always been a goal of the author's. The author thinks that this project is very close to what could be a real world scenario - for a business to hire someone to build software using open source and a hosting company to host the application versus hiring permanent IT staff for small projects. The level of quality and support from a good hosting company such as WebAppCabaret.com, is just as good, if not better, and more cost effective than hiring and creating a complete IT department for a small project or company.

What You Would Have Done Differently in the Project

The author would like to have had more time to research the different technologies and hosting companies. The author felt like he made a good decision on using the open source projects, but since completing most of the software, the author has learned there are several better or similar technologies that might have been easier and faster to implement that provided the same capabilities. When the author looked back on the project, the prototype and agile methodology was good and helped identify critical path issues, but maybe spending time in a formal design phase might have helped the project spend less time re-coding on

certain areas. The author also thinks that getting some UI specification documents built using a tool such as Visio would have helped lay out the site prior to building the actual code, and would also have allowed the customer to understand where the project was headed. It was not until the code was available on the hosting company's server, that he got to see what was built and how the system would interact. From a project management perspective, the author thinks he could have laid out the time required by the project and planned accordingly. The author had the requirements documented and kept coding until all the requirements were completed. The author thinks if he had the actual project plan down to a task level, then he could have seen the remaining work and been able to prioritize the time spent on the project.

Discussion of Whether the Project Met Expectations

The author felt the project met the original expectations. First, the customer was fine with this taking a bit longer and agreed to be flexible as the author was a part-time student and not a full-time dedicated resource. The original request from the client was to get the PAIR test working online. But it did not include the UI interface rewrite or the fact that a new hosting company had to be evaluated and selected. The initial project scope from the customer did not take into account all the administrative functions needed to be able to manage users, counselors and questions for the tests. This was all additional work that was required for a minimal product to make the base requirement of providing the test online. The

project did deliver the working system and at this present time there are fifteen beta test clients using the system. The customer has reported that the online version is saving about 65 percent of time for the entire process. Couples are completing the test in about 20 minutes versus 60 in the offline process. Counselors used to spend about 1-1.5 hours scoring the results and now can do that in about 15 minutes. Therefore, the project has already shown significant increases in productivity using the online process in comparison to the offline version. The time for scoring results will be reduced even further once the future enhancements are completed. One of the future enhancements is to complete the automatic plotting of the profile. Currently, the counselors get automatic scoring, but have to transfer that to the hard copy profile form. The author and the customer de-scoped this to spend time on making the delivery and getting the quality around the test.

Evolution for the Project/Next Steps

Maintenance

The project needs development and deployment documents and the code tools made available so that another developer could begin to enhance the system. Currently, there are aspects of the code and deployment that have limited documentation. The documentation was identified, but was unable to be completed because of time and scope. Currently, the author would like to document some of the development

and deployment procedures, but there is no time available for that effort for this project. Therefore, it is the assumption of the author that when and if this project continues with another developer/student, the author will help transfer and train this developer/student on the development environment and deployment process, as well as general site maintenance for the hosting account with WebAppCabaret.

The hosting company provides the capability to perform daily data backups of the database. This should be explored and the proper steps taken to ensure that is enabled and working. These steps are for retrieving the archived data and information. When the last successful backup was performed should also be investigated by a follow-up project. This would be important for the customer to understand so proper communications could be made to the users of the system in the event the data in the database is lost.

Another item that needs to happen, that has not already been completed, is training the customer how to use the hosting company's management/control panel. This management application is used to do web site domain management to data manipulation in the database. This is an important piece for the customer in taking ownership of the application and site past this project.

Enhancements

Currently, couples are treated as completely separate people and not grouped as a pair. This makes payments required for both, and does

not allow counselors to be able to view results for the couple by last name. The counselor must know both of their user id's. Tying the users' accounts together would make it easier to process payments and view results. Also, the system could use a notification process for several items, such as notifying a counselor once both persons have completed the tests and sending an email with the PDF.

There is also no reporting of the profile data and the types of responses so that the data can be used as a profiling and relationship analytics tool. There could also be an enhancement to finish off the relationship profile PDF to have the lines generated as an image as well. This was originally in scope, but removed based on time and effort.

The system could also tie in a scheduling function to allow counselors to set up appointments via the web site to review and discuss the findings of the profile results. This is where the benefit of all the testing and scoring is used. The counselor uses the profile results during a counseling session to discuss personality traits that the couples have that will cause issues in a marriage or relationship.

Conclusions / Recommendations

The project met the goals established early on to build an online application that provides the same functionality currently offered in the offline version of the PAIR Test. This project contributed to the foundation, framework and base for which the PAIR Test can continue to grow and advance. Using open source technologies will allow the

customer to possibly have more students in future projects extend this application without extensive cost or knowledge of proprietary technologies. Open source is the fastest growing technology area today. Therefore, the ability to find developers and/or students should not be an issue in the future. A few recommendations are to have the customer use the system for a few months to find out how the system performs and functions, then begin planning the next releases for how to make the system better. Many times people try to decide how systems will work best and really don't know until they are used by a wide variety of audiences with different perspectives. These new perspectives will help define process improvements and new ways of looking at the solution to make it better.

Summary

This project was very challenging, but has also been one of the most rewarding experiences. It offered the author many learning opportunities that would not have been possible without this project. The author is very grateful for the chance to be part of creating something meaningful from scratch. This project has given the author experience in the following areas

- Project management lessons on defining scope and establishing a good methodology for planning and executing on the plan are invaluable lessons to a developer that just writes code or manages a small team. This really helps

developers contribute to a project much more effectively and the author now values the role of a project manager much more than prior to this project.

- Research and comparison of technology solutions is something that does not happen as often as it should. This project allowed the validation that just because a technology is new and/or exciting does not make it the best solution for an implementation. More times than not, we, as technologists, pick technologies for reasons based on pressure from outside factors and not based on what the real needs are and use what works and is the least risk and mostly likely to succeed. This project could have been just as successful in its goal if Hibernate was not used. However, the author felt learning Hibernate and using Hibernate would save time on coding the database access code. This was not the case - the Hibernate code actually took more time and eventually was removed from the project. However, with that said, the author felt that the time spent learning Hibernate was valuable and offers him a better understand of this product. The next time the author will try to examine new technologies as a separate element than the project. He would then include the elements for a project based on risk factors, timelines and what is needed

for the project to reach its goals versus popularity or other factors that don't contribute to the goals of the project.

- Using Agile/XP systems development methodologies has proven to be effective in projects that have small dynamic teams. This method was appropriate for this project and worked well, but using Agile/XP on a large project with many varying skill sets could be hard to manage. The construction phase that followed the analysis and design phase had better quality code and the time to develop was much shorter than the prototype phase coding efforts. This validates that projects can and should adapt to and use methods that work best for different points of a projects duration.
- Researching and choosing a hosting company for this project was an element that the author did not account for being a major piece. However, that was hugely under valued by the author. The author selected the first hosting company based on the fact that it was inexpensive, could host java applications and had support for the MySQL database server. The author realizes now that this piece of the project was a major element of the pre-project research that should have been taken more seriously. Hosting companies all offer different services and value to their

customers. Having a hosting company that provides application level support is more useful than saving cost. The first hosting company's cost was $9/month and second was $17/month. The level of the final hosting company provided great support for java applications and staff that had java experience. A good hosting company that supports the technologies selected for an application is just as important to the success of a project as the scope, resources, and project management.

This project incurred about a three month delay. The majority of this project's delay was because of the new technologies and the author's first time developing a production quality product from the ground up and being without help in many areas. The selection of the hosting company was the most under analyzed element of the project. The biggest take away from this project is to really understand, research and analyze case studies of existing solutions during the pre-project of any project will help reduce risks. Use this information to try to minimize bad decision making in your own projects and this will make future projects more efficient and reduce risk.

Appendix – References

Apache Struts (n.d) Retrieved September 2005 from http://struts.apache.org/

Brown, J., (n.d) An Introduction To Java Data Objects, Object Computing, Inc. (OCI) (http://www.ociweb.com/jnb/jnbJun2002.html

Complete Hibernate 3.0 Tutorial (n.d) Retrieved September 2005 from http://www.roseindia.net/hibernate/index.shtml

CSS design benefits (n.d.) Retrieve September 2005 from http://www.webcredible.co.uk/benefits/css-design.shtml

CQuest Assessment Software (n.d) Retrieved October 2005 from www.cquestsoftware.com

Hanson, J., (2004). Simplify Java Object Persistence with Hibernate Retrieved September 2005 from Http://www.devx.com/Java/Article/22652

HIBERNATE - Relational Persistence for Idiomatic Java (n.d) Retrieved September 2005 http://www.hibernate.org/hib_docs/v3/reference/en/html/

Hibernate Relational Persistence for Java and .NET (n.d.) Retrieved September 2005 from http://www.hibernate.org/

Husted, T., Dumoulin, C., Franciscus, G., Winterfeldt, D., (2003) Struts in Action, Building Web Applications with the Leading Java Framework, Mannig press.

Lowagie, B., Soares, P., (n.d) iText - A Free PDF Library Retrieve September 2005 from http://www.lowagie.com/iText/docs.html

Malani, P., (2002) UI design with Tiles and Struts Retrieved from http://www.javaworld.com/javaworld/jw-01-2002/jw-0104-tilestrut.html

MyEclipse J2EE IDE (n.d) Retrieved September 2005 from http://www.myeclipseide.com/

MySQL Reference Manual (n.d.) Retrieved September 2005 from http://dev.mysql.com/doc/refman/5.0/en/

Perception Online Assessment Software (n.d) Retieved August 2005 from http://www.pearsonncs.com/home.htm

Perception by QuestionMark Inc. (n.d.) Retrieved August 2005 from http://www.questionmark.com/us/home.htm

Open Source Software in Java(tm) (n.d) Retrieved August 2005 from http://www.java-source.net/

Richmond, A., (n.d.) Benefits of Cascading Style Sheets Retrieve September 2005 from http://www.wdvl.com/Authoring/Style/Sheets/Benefits.html

Singh,I., Stearns, B., Johnson, M., and the Enterprise Team (2002). Designing Enterprise Applications with the J2EETM Platform, Second Edition Retrieved September 2005 from http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/DEA2eTOC.html

Shannon, R., (n.d.) Introduction to CSS Retrieved September 2005 from http://www.yourhtmlsource.com/stylesheets/introduction.html

Scheinblum, J., (2002) Make your Applications Strut Retrieved September 2005 from http://techrepublic.com.com/5100-22_11-1027640.html?tag=search

Appendices

Appendix A: Use Case Diagrams

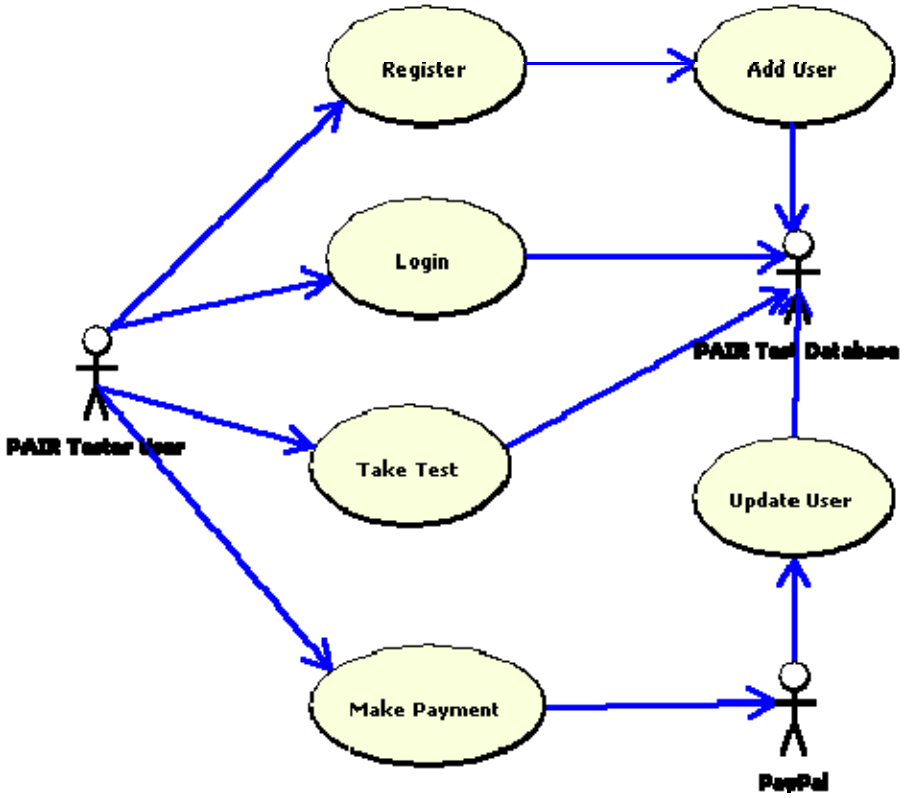

Figure 3. *PAIR Test tester use case model.*

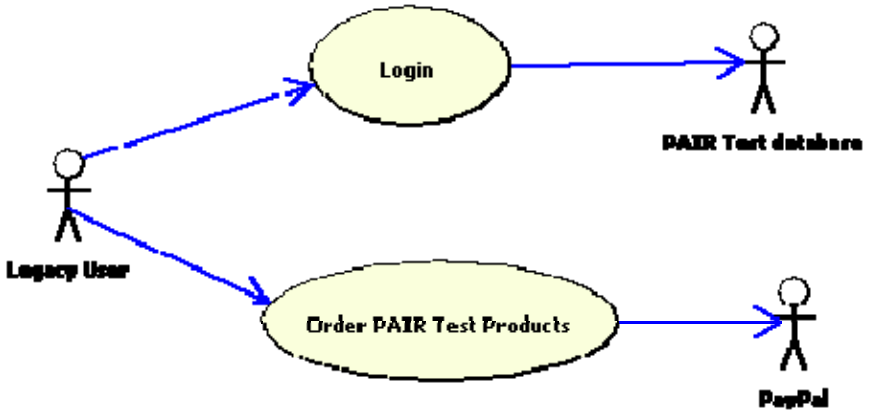

Figure 4. *Legacy Counselor user use case model.*

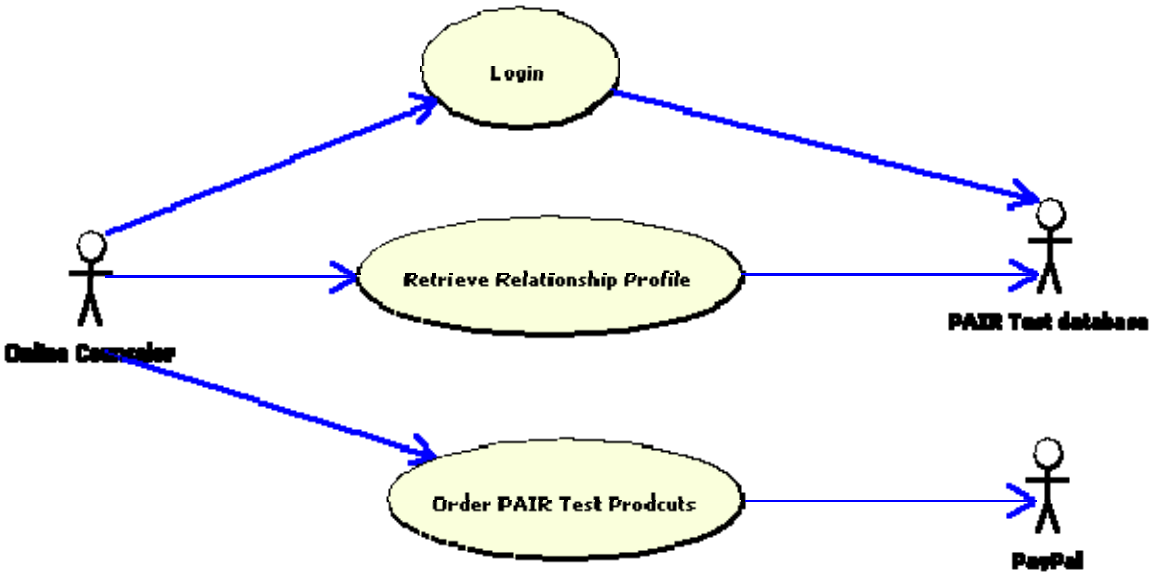

Figure 5. *Online Counselor user use case model.*

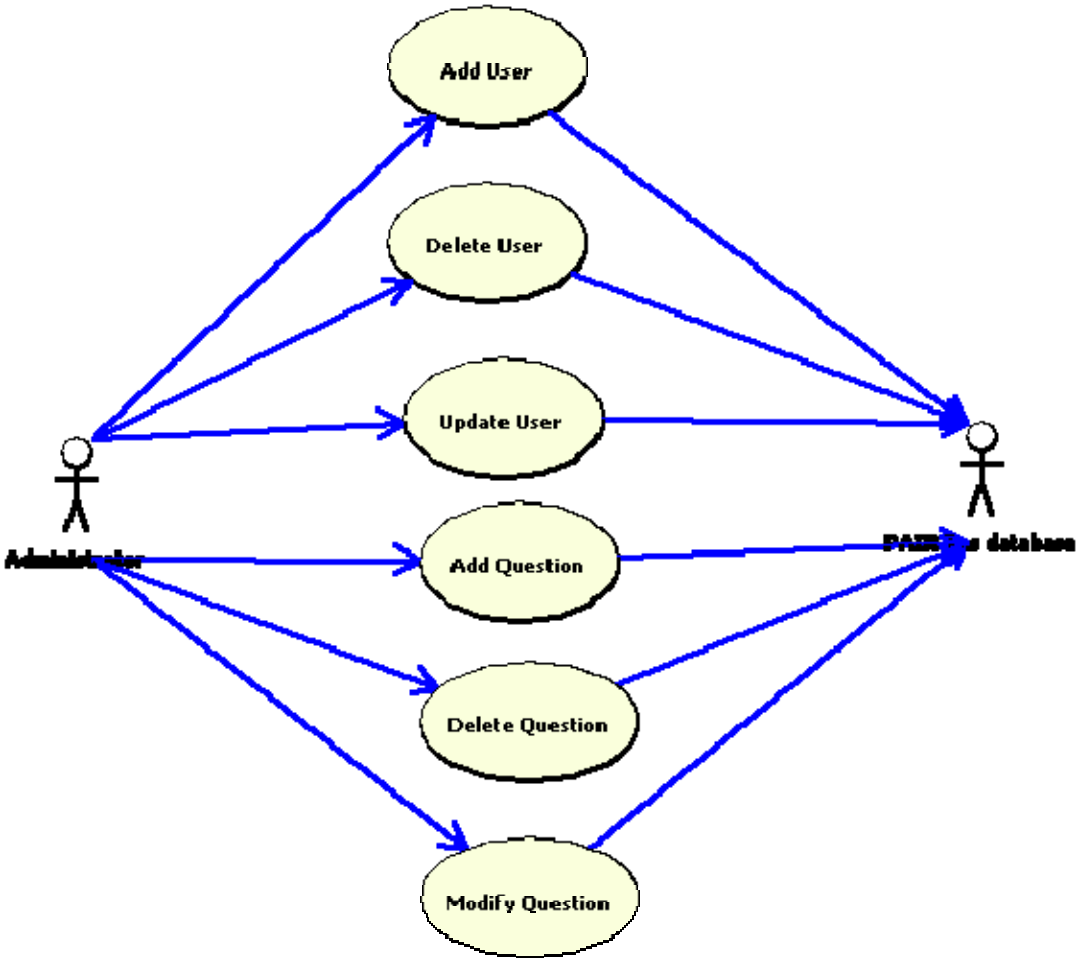

Figure 6. *Admin user use case model.*

Appendix B: Use Case Template

Use Case Template – User Self Register

Use Case ID:	U100		
Use Case Name:	User Self Registration		
Created By:	Kevin Hayes	Last Updated By:	Kevin Hayes
Date Created:	3/11/06	Date Last Updated:	3/11/06

Actors:	Online Pair2 Patrons
Description:	Self Registration and User Account Set-up, this is the process to create a Pair2User account to be used to allow test completion and results tracking.
Preconditions:	1. The user has navigated to the Self registration page and has been given the counselor Id to create an account.
Post Conditions:	1. User Account has been created 2. User has been sent to the make payment page to complete the payment part of the registration.

Normal Flow:	1. User completed all the data elements of the User Registration Page and clicks the submit button.
Alternative Flows:	1. N/A
Exceptions:	Fix all page errors for the form elements.
Includes:	
Priority:	
Frequency of Use:	
Business Rules:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Appendix C – Login JSP code

login.jsp (page definition)

```
<%@ page language="Java"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-tiles" prefix="tiles" %>
<tiles:insert page="../../layout/mainPageLayout.jsp">
        <tiles:put name="pageTitle" value="The PAIR Test"/>
        <tiles:put name="bodyContent"
value="../module/test/purchaseLoginModule.jsp"/>
        <tiles:put name="header" value="../module/global/header.jsp"/>
        <tiles:put name="footer" value="../module/global/footer.jsp"/>
</tiles:insert>
```

layout.jsp

```
<%@ page language="Java"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-tiles" prefix="tiles" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html locale="true">
```

```
 <head>
  <html:base/>
  <title><tiles:getAsString name="pageTitle" /></title>
  <meta http-equiv="pragma" content="no-cache">
  <meta http-equiv="cache-control" content="no-cache">
  <meta http-equiv="expires" content="0">
  <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
  <meta http-equiv="description" content="This is my page">
  <link href="<%=request.getContextPath()%>/css/pair.css" rel="stylesheet"
type="text/css">
 </head>
 <body>
        <div id="bodyContainer">
              <div id="headerContainer">
                    <tiles:get name="header"/>
              </div>
              <div id="bodyContentContainer">
              <tiles:get name="bodyContent"/>
        </div>
        <div id="footerContainer">
           <tiles:get name="footer"/>
              </div>
  </div>
 </body>
</html:html>
```

header.jsp – view component

```
<%@ page language="Java"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
        prefix="html"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"
        prefix="logic"%>
<%@ taglib uri="http://Java.sun.com/jstl/core" prefix="c" %>
<%@ page import="org.apache.struts.action.Action" %>


<img src="<%=request.getContextPath()%>/images/couple-on-the-beach.jpg" alt="Pair
Test" title="Logo" />
<%--<img src="../images/couple-on-the-beach.jpg" name="Logo" alt="logo name">--%>
<table>
        <tr>
              <td><logic:notPresent name="client">
                    <html:form action="/purchaseLogin">
                          <table>

                                <tr>
                                      <td>User Name</td>
                                      <td><html:errors property="userName"
/><html:text
                                            property="userName" /></td>
                                </tr>
                                <tr>
                                      <td>Password</td>
```

```
                                                <td><html:errors property="password"
/><html:password
                                                        property="password" /></td>
                                        </tr>
                                        <tr>
                                                <td></td>
                                                <td><html:submit /></td>
                                        </tr>
                                </table>
                        </html:form>
                </logic:notPresent> <logic:present name="client">
                        <html:link action="/logout.do">Log Out</html:link>
                </logic:present></td>

        </tr>
</table>

<c:choose>
        <c:when test="${selectTab == 'index'}">
                <c:set var="indexClass" value="current"/>
        </c:when>
        <c:when test="${selectTab == 'factsheet'}">
                <c:set var="factsheetClass" value="current"/>
        </c:when>
        <c:when test="${selectTab == 'video'}">
                <c:set var="videoClass" value="current"/>
        </c:when>
        <c:when test="${selectTab == 'purchase'}">
                <c:set var="purchaseClass" value="current"/>
        </c:when>
        <c:when test="${selectTab == 'contact'}">
                <c:set var="contactClass" value="current"/>
        </c:when>
        <c:when test="${selectTab == 'registration'}">
                <c:set var="registrationClass" value="current"/>
        </c:when>
        <c:when test="${selectTab == 'approvedPurchase'}">
                <c:set var="approvedPurchaseClass" value="current"/>
        </c:when>
        <c:when test="${selectTab == 'profileRequest'}">
                <c:set var="profileRequestClass" value="current"/>
        </c:when>
        <c:when test="${selectTab == 'adminHome'}">
                <c:set var="adminHomeClass" value="current"/>
        </c:when>
        <c:when test="${selectTab == 'takeTest'}">
                <c:set var="takeTestClass" value="current"/>
        </c:when>
        <c:otherwise>
        </c:otherwise>
</c:choose>


<div id="headerItems">
```

```
<ul>
		<li id="<c:out value='${indexClass}' />"> <a
href="<%=request.getContextPath()%>/home.do">Home</a> </li>
		<li id="<c:out value='${contactClass}' />"> <a
href="<%=request.getContextPath()%>/contact.do">Contact</a></li>

		<logic:present name="client">

			<logic:equal name="client" property="usertype" value="pair2user">
				<logic:equal name="client" property="paymentreceived"
value="t">
					<logic:notPresent name="testComplete">
						<logic:notPresent
name="<%=Action.ERROR_KEY%>">
							<li id="<c:out value='${takeTestClass}'
/>"><a href="<%=request.getContextPath()%>/question.do">Take Test</a></li>
						</logic:notPresent>
					</logic:notPresent>
				</logic:equal>
			</logic:equal>
			<logic:equal name="client" property="usertype" value="legacyuser">
				<li id="<c:out value='${factsheetClass}' />"> <a
href="<%=request.getContextPath()%>/pagedef/factsheet.jsp">Fact Sheet</a></li>
				<li id="<c:out value='${videoClass}' />"> <a
href="<%=request.getContextPath()%>/pagedef/video.jsp">Video/DVD</a></li>
				<li id="<c:out value='${purchaseClass}' />"> <a
href="<%=request.getContextPath()%>/pagedef/purchase.jsp">Purchase</a></li>

			</logic:equal>
			<logic:equal name="client" property="usertype" value="counselor">
				<li id="<c:out value='${factsheetClass}' />"> <a
href="<%=request.getContextPath()%>/pagedef/factsheet.jsp">Fact Sheet</a></li>
				<li id="<c:out value='${videoClass}' />"> <a
href="<%=request.getContextPath()%>/pagedef/video.jsp">Video/DVD</a></li>
				<li id="<c:out value='${purchaseClass}' />"> <a
href="<%=request.getContextPath()%>/pagedef/purchase.jsp">Purchase</a></li>
				<li id="<c:out value='${profileRequestClass}' />"><a
href="<%=request.getContextPath()%>/pagedef/counselor/counselorprofilerequest.jsp">
Retrieve
				Profile</a></li>
			</logic:equal>
			<logic:equal name="client" property="usertype" value="adminuser">
				<li id="<c:out value='${factsheetClass}' />"> <a
href="<%=request.getContextPath()%>/pagedef/factsheet.jsp">Fact Sheet</a></li>
				<li id="<c:out value='${videoClass}' />"> <a
href="<%=request.getContextPath()%>/pagedef/video.jsp">Video/DVD</a></li>
				<li id="<c:out value='${purchaseClass}' />"> <a
href="<%=request.getContextPath()%>/pagedef/purchase.jsp">Purchase</a></li>
				<li id="<c:out value='${adminHomeClass}' />"> <a
href="<%=request.getContextPath()%>/pagedef/admin/adminhome.jsp">Admin
Home</a></li>
			</logic:equal>
		</logic:present>
		<logic:notPresent name="client">
			<li id="<c:out value='${registrationClass}' />"><a
href="<%=request.getContextPath()%>/pagedef/test/registration.jsp">Register</a></li>
```

```
        </logic:notPresent>
</ul>
</div>
```

loginModule.jsp – view component

```
<%@ page language="Java"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
        prefix="html"%>
<html:errors/>
<html:form action="/purchaseLogin">
        <table>
                <tr>
                        <td>*User Name</td>
                        <td><html:errors property="userName" /><html:text
property="userName" />
                        </td>
                </tr>
                <tr>
                        <td>*Password</td>
                        <td><html:errors property="password" /><html:password
property="password" />
                        </td>
                </tr>
        </table>
        <html:submit />
        <html:cancel />
</html:form>
```

footer.jsp – view component

```
<%@ page language="Java"%>
```