Designing & Implementing a Database Driven Web-Based Volunteer Management System

Jonathan Franklin Spencer
&
Adam Tyler Huffman

April 10, 2006

Designing & Implementing a Database Driven Web-Based Volunteer Management System

Jonathan Franklin Spencer
&
Adam Tyler Huffman

Technical Report

A Senior Project Report
Submitted to the Computer Science Faculty
of Millsaps College
in Fulfillment of the Requirements
for the Degree of
Bachelors of Science in Computer Science

Project Advisor - Dr. Donald Schwartz

Millsaps College
Jackson, MS 39210
April 10, 2006

Abstract

Following the influx of volunteers after Hurricane Katrina, the Central Mississippi Chapter of the American Red Cross contacted the Department of Computer Science for help managing its volunteers. The result was a system codenamed “Canadian Blue Minus”, which was implemented using PHP and MySQL, developed by senior computer science majors Adam T. Huffman and Jonathan F. Spencer under the guidance of Dr. Donald R. Schwartz. This application met the needs of the Red Cross, but the research group decided to expand the project to handle more of the operations of non-profit organizations, such as client, location, and employee management. A more robust implementation was desired as well, so the group chose Java Server Faces to interface with the database, for which they continued to use MySQL. Because Java Server Faces is a very new technology, the project suffered many setbacks. Despite these, the group managed to gain a vast amount of knowledge concerning Java web technologies and database design. A working server that hosts the application for the Red Cross has also been built and is soon to be delivered.

Key Words : Sun Microsystems Java, Java Server Faces, Java Server Pages, Java Servlets, J2EE, Sun Java System Application Server, MySQL, PHP, Apache Tomcat, Sun Java Studio, CVS, Samba, RedHat Fedora Core 4, web technologies, volunteerism, volunteer management, non-profit organization, Hurricane Katrina

Contents

1 Introduction **1**

2 Preliminary Work: the Prototype **2**

2.1 System Requirement 3 2

2.2 Application User Roles 3

2.2.1 Volunteer 3

2.2.1.1 Registration 3

2.2.1.2 My Info 4

2.2.1.3 Update 4

2.2.1.4 Contact Administrator. 4

2.2.2 Administrator 5

2.2.2.1 Administrator Queries. 5

2.2.2.1.1 Contact Wizard 6

2.2.2.2 Worker Management 6

2.2.2.2.1 Register 6

2.2.2.3 System Management 7

2.2.2.4 Message Center 7

2.3 Implementation 7

2.3.2 Linux 2.6.11-1.1369 8

2.3.2.1 Fedora Core 4 8

2.3.2.2 Samba Server 3.0.14a 8

2.3.3 Apache HTTP Server 2.0.54 8

2.3.4 MySQL 4.1 8

2.3.4.1 MyISAM 9

2.3.4.2 Database Entity Relationship Diagram 9

2.3.4.3 MySQL Administrator and Query Browser 10

2.3.5 PHP 5.0.4 10
2.3.5.1 Sessions 10
2.3.5.2 Includes 10

3 Evaluation of the Prototype **11**
3.1 Reusable Modules 11
3.2 Generic Design 11
3.3 Security Issues 12
3.4 Missing Functionality 13

4 iCare **15**
4.1 Sparta – Web Application Server 16
4.1.1 Hardware 16
4.1.2 Network 16
4.1.3 Operating Systems 16
4.1.3.1 Sun Solaris 10 17
4.1.3.2 Fedora Core 4 17
4.1.3.2.1 Network 17
4.1.3.2.2 Users 18
4.1.3.2.3 TTY Over Null Modem Cable 18
4.1.3.2.4 SELinux 18
4.1.4 Software 19
4.1.4.1 Samba Server 19
4.1.4.2 Java 5 20
4.1.4.3 CVS 1.11.21 20
4.1.5 Application Servers 20
4.1.5.1 Tomcat Server 5.5.12 21
4.1.5.2 Sun Java System Application
Server Platform Edition 8.2 22

4.2 Athens – Database Server 22
4.2.1 Building the Server 22
4.2.2 Installing the Operating System 23
4.2.3 Installing and Configuring MySQL 5.0 24
4.2.4 Creation and Optimization of the Database 24
4.3 Development Machines 27
4.3.1 Hardware 27
4.3.2 Software 28
4.3.2.1 Operating Systems 28
4.3.2.1.1 Windows XP Professional SP2 28
4.3.2.1.2 Fedora Core 4 28
4.3.2.2 Integrated Development Environments 29
4.3.2.2.1 Borland JBuilder 2005 29
4.3.2.2.2 NetBeans 4.1 29
4.3.2.2.3 Eclipse 3.1.2 29
4.3.2.2.4 Macromedia Dreamweaver 29
4.3.2.2.5 Sun Java Studio Creator 2 30
4.3.2.2.6 Sun Java Studio Enterprise 30
4.4 Java Server Faces 30
4.4.1 Proposed Final Draft 30
4.4.2 Compiling the Pages 31
4.4.3 Navigation 31
4.4.4 Layout Issues 32
4.4.5 Resolving Errors 33
4.5 Java Studio Creator 2 33
4.5.1 CVS 34
4.5.1.1 Add 34
4.5.1.2 Update 34
4.5.1.3 Commit 34
4.5.1.4 Merge 35
4.5.2 System Requirements 35

4.5.3 Handling of Deletion 35
4.5.4 Documentation Not Yet Published 36
4.5.5 Updates Break Workarounds 36
4.5.6 Random Bean Placement 36
4.5.7 Sun Java Studio Application Server Platform Edition 37
4.6 Sun Java System Application Server 8.2 37
4.6.1 Documentation 38
4.6.2 Connection to MySQL 38
4.6.3 Deployment 39
4.6.4 Bugs 39

5 Conclusion 40

References 41

Appendices 42
A Development References 42
B Retrospect 44
C Future Work 45

1 Introduction

Non-profit organizations have quite a challenge: serve the most people possible with the least funding. To do so, organization managers must run tight ships; they must find ways to spend most of their money on the people they serve and not the business operations that support them. For this reason, computer systems at non-profit organizations are typically older, possibly donated machines that nobody really has time to upgrade or setup to serve the people who are serving the general public.

The Central Mississippi Red Cross was no exception. Despite its federal funding, the computer systems available to the Red Cross served only basic, generic purposes—email, office applications, and website hosting. The Red Cross handled volunteer information the old-fashioned way, by storing pieces of paper in filing cabinets. If a client could not speak English, a volunteer would search through thousands of applications by hand to find a volunteer who could translate and hope that he or she was available and could be contacted using the information on the form, which could be years old. This system was very low-tech, easy to learn, and did not cost much to implement, except time.

When Hurricane Katrina ransacked the Gulf Coast in August, 2005, time ran out. Red Cross Shelters swarmed not only with people escaping the catastrophe, desperately needing aid, but also with people who desperately wanted to help. This was no time for paper forms; as soon as clients and volunteers would fill them out, they would be lost in the confusion or lost in the already uncontrollable collections in the filing cabinets. To compound the matter, there was more work than any amount of volunteers could handle, and none could be spared to sort through piles of papers. The Red Cross made attempts to have volunteers enter the data into spreadsheets, but the lack of consistency among spreadsheets and the ease with which spreadsheet data can be lost became more of an aggravation than an answer.

To remedy their situation, the Central Mississippi Red Cross contacted the Computer Science Department at Millsaps College. Dr. Donald Schwartz, chair of the department, suggested he have some of his students build a system as a service learning project [1]. Senior computer science majors Adam Huffman and Jonathan Spencer accepted this challenge as the curriculum for CSCI 3752 Advanced Topics in Database. The result of this class was a working implementation using PHP and MySQL, codenamed "Canadian Blue Minus", which they demonstrated for the Red Cross in December, 2005 [2]. As their senior project, a requirement for graduation from the Department of Computer Science at Millsaps College, Adam and Jonathan decided to improve upon the current system and build a working server to host the application [3], which they would deliver to and setup for the Central Mississippi Red Cross. The improved system, codenamed "iCare", would be implemented using Java Server Faces and MySQL [4].

2 Preliminary Work: the Prototype

"Canadian Blue Minus" is the codename for the prototype system. It implements most of the requirements from the Red Cross using technologies that lend themselves to short development periods. The main purpose of the Canadian Blue Minus project was to produce a usable system in a very short amount of time that demonstrated the capabilities of a web-based, database-driven volunteer management system.

2.1 System Requirements

The Red Cross' first and foremost requirement was that the system would be consistent; that is, that data is always entered and retrieved in the same manner. This helps avoid data loss due to redundancy and omissions when several volunteers are working on the same task.

Next, the system had to have a way to quickly search through all the volunteers to find those who met certain criteria. The Red Cross needed to find volunteers who lived in certain areas, had specific skills and certifications, and had worked on previous events.

They also wanted to know at what times their volunteers were available to work, whether they had their own transportation, whether they had ever been convicted of felonies, etc. "Thank You" and "We Miss You" letters should also be generated for volunteers who worked recently or have not worked in a specified amount of time.

Another requirement was that the system had to be accessible to many people simultaneously from different locations over the Internet. Inherent in this requirement was the need for user roles and accounts for different levels of authorization within the system. The Red Cross also wanted some way for non-connected computers in the field to log volunteer work histories and later merge that data with the master database.

2.2 Application User Roles

The Canadian Blue Minus system has two user roles, volunteer and administrator, to handle separate authorization levels for users with different privileges.

2.2.1 Volunteer

A volunteer is a person who is willing to and/or performs work for the Red Cross. A volunteer has a core set of contact and identification information, may posses many of several skills and certifications, may be affiliated with many of several organizations, etc.

2.2.1.1 Registration

Registration is a replacement for the volunteer information paper forms that volunteers currently complete. After completing this process, a volunteer has a user account with the Canadian Blue Minus system and has inserted her information into the database.

The set of required information consists of the user's first and last name, street address, city, state, zip code, unique email address, date of birth, gender, and a password. The user can use her email address and password to login to the system from any computer connected to the Internet.

The subsequent screens collect extra information about the volunteer. This includes the user’s skills, classes she would like to take, languages that she speaks, emergency contact, etc. This also includes the times that the user is available to volunteer, divided into days, then morning, midday, and night, and finally into thirty minute time slots.

If the user has questions about the registration process or needs some option changed (such as an organization added to the list), she can click “Contact Administrator” at any time. This will bring up a form with information relevant to the current section and send it to the administrators upon submission [3].

2.2.1.2 My Info

Upon logging into the system as a registered user, a volunteer may click the “My Info” button to see a complete listing of her information as recorded by the Canadian Blue Minus system. This includes her personal information, availability, affiliations, certifications, skills, classes, and work history. If a volunteer needs a receipt for working at a certain event to fulfill a philanthropic requirement, she may simply login to the system and view this page [3].

2.2.1.3 Update

From the My Info section, a user may click the “Update” link from any segment to add or change information as she pleases. This brings up a form that allows direct input by the user to modify the current information in the system [3].

2.2.1.4 Contact Administrator

Some aspects of the My Info section should not be changed by the user, such as the listing of certifications and work history. When the user clicks the Update link for these sections, she is given a form to contact the system administrators with the necessary information and automatically attach her email address and the time of the message [3].

2.2.2 Administrator

An administrator is a volunteer who has been granted privileges to make modifications and updates to the system by another administrator. An administrator also inherits all the privileges and has access to all the features of a volunteer [3].

2.2.2.1 Administrator Queries

The administrator queries section allows administrators to search for volunteers based on their chosen criteria and generate statistics based on the current volunteer information in the database. Sample queries include:

- List volunteers from a certain city
- List volunteers and their hours for a certain date/event
- List volunteers and their hours as part of a certain organization for a certain event/date
- List volunteers who have received a certain certification
- List volunteers who have not been contacted in a defined period of time
- List volunteers who are available at a particular time
- List the number of volunteers who have (not) completed each training
- List the average age of volunteers
- List volunteers who have not worked in a defined period of time
- List volunteer who recently worked
- List volunteers whose information recently changed
- List volunteers who desire an upcoming course

The last four queries return the information necessary to send letters to volunteers for “We Miss You” and “Thank You” letters as well as general notification letters [3].

2.2.2.1.1 Contact Wizard

The administrator queries section gives administrators a fast and easy way to find volunteers and solicit their help. Using this section of the application replaces searching through filing cabinets and reading thousands of potentially out-of-date forms to find aid.

Each section of the administrator queries that returns a list of volunteers has the option to format the results for contacting. This adds an input field and contact summary to each resulting volunteer’s information. To contact a volunteer, an administrator tries one of the contact methods listed, then records the results of the contact session in the text field and submits the form.

Each contact summary contains the name of the administrator who contacted the volunteer, the time of the contact session, and the details of the session. The first two fields are captured automatically when the administrator submits the details of the session [3].

2.2.2.2 Worker Management

Once a work session is over, the Canadian Blue Minus system makes it easy to enter the results into the system. The worker management section contains the same information as a sign in sheet, such as volunteer in/out times, the skill they used, the organizations with which they were working, and the event [3].

2.2.2.2.1 Register

If an administrator tries to enter information for a volunteer who is not a registered user of the system, the administrator is prompted to setup a temporary account that contains the core information for a volunteer [3].

2.2.2.3 System Management

As the number of volunteers grows, some information will need to be updated or added to keep the system current. The system management section allows administrators to add and update certifications, courses, events, languages, and skills, as well as grant (or take away) administrative privileges to other users [3].

2.2.2.4 Message Center

When administrators login to the system, they have the option of checking answered and unanswered messages from the system users.

The unanswered messages section lists questions and requests from users that have not yet been handled. These messages include information about the request and the user who posted the request, as well as an input field. Upon answering the question or handling the request, the administrator types a summary of her actions in the text field and submits the response.

Requests that have already been handled appear in the list of answered messages. Along with each question is the response by the administrator who handled it, the name of the administrator, and the time that the response was submitted [3].

2.3 Implementation

Since the Red Cross is a non-profit organization, the system is implemented using free/open source software to keep the cost of the system as low as possible. The combination of technologies is commonly referred to as L.A.M.P., short for Linux, Apache, MySQL, and PHP [4].

2.3.2 Linux 2.6.11-1.1369

Linux is an open source operating system originally written by Linux Torvalds based on the POSIX operating system. It provides a great deal of flexibility while still maintaining security and stability.

2.3.2.1 Fedora Core 4

Fedora Core 4 is a Linux distribution jointly maintained by the Fedora Community and RedHat. It strives for user-friendliness and stability to push Linux into the mainstream.

2.3.2.2 Samba Server 3.0.14a

Samba Server allows *nix-based machines to mimic the functions of Windows-based short message block network servers. The Canadian Blue Minus server uses Samba Server to host a network share that gives certain users access to the contents of the website for remote editing and deployment.

2.3.3 Apache HTTP Server 2.0.54

Apache HTTP Server is an open source web server that runs on most operating systems, including Linux. It allows easy and powerful interoperability with PHP and MySQL.

2.3.4 MySQL 4.1

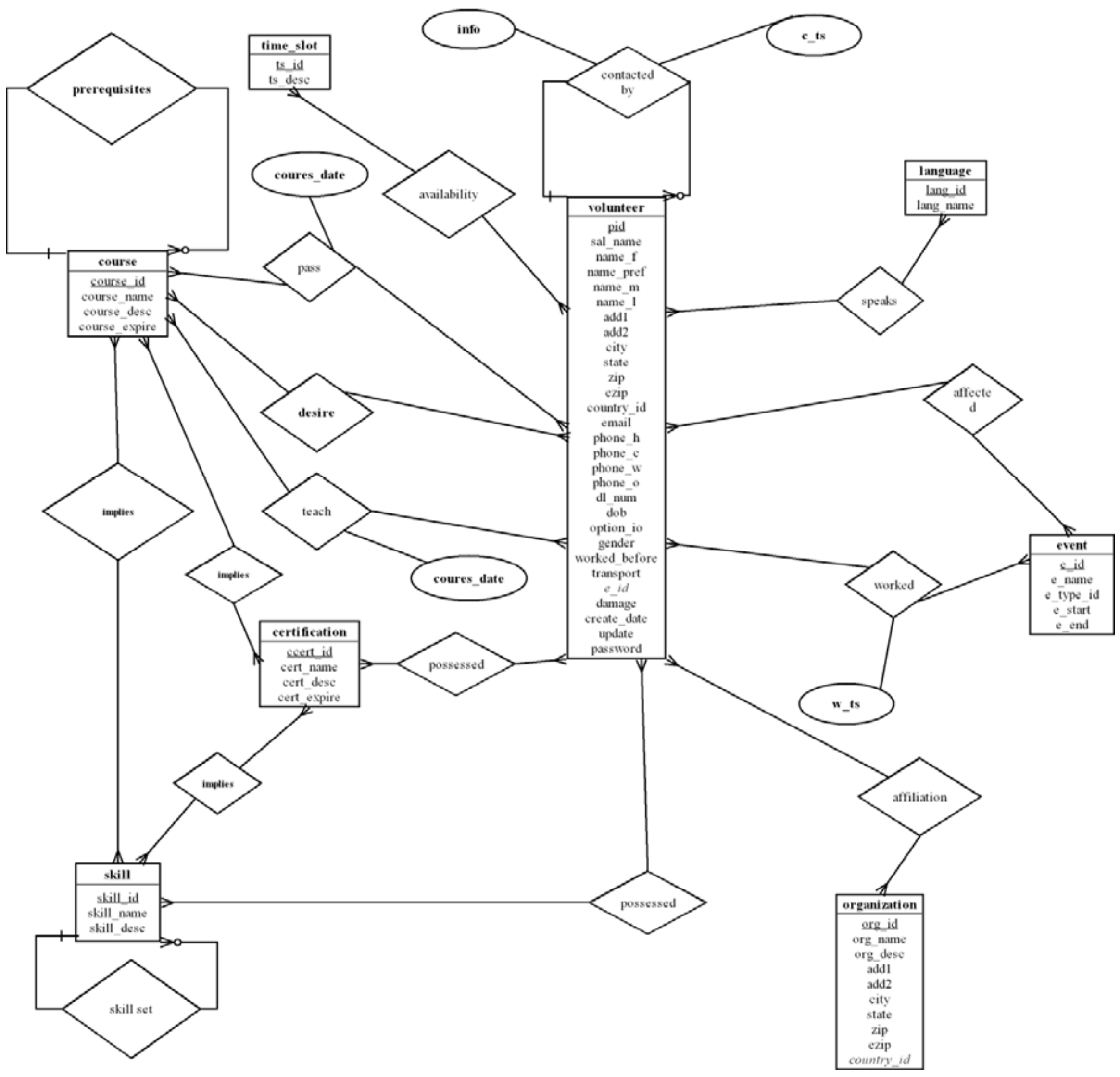
MySQL 4.1 is a relational database management system designed by the Swedish company MySQL AB. MySQL 4.1 is available as free software under the GNU General Public License (GPL), but they also dual-license it under traditional proprietary licensing. At the time of system design MySQL 4.1 was the most stable and documented version available.

2.3.4.1 MyISAM

MyISAM is the default storage engine for MySQL. It is based on the older IBM Indexed Sequential Access Method (ISAM). Upon creation of a MyISAM table MySQL automatically selects the table to be dynamic or static if the table does not contain and VARCHAR, BLOB, or TEXT columns. Most tables contained within Canadian Blue Minus are dynamic.

2.3.4.2 Database Entity Relationship Diagram

Canadian Blue Minus ER-Diagram



2.3.4.3 MySQL Administrator and Query Browser

MySQL Administrator and MySQL Query Browser are free utilities provided by MySQL AB. MySQL Administrator is a GUI for editing the total schema of a MySQL database system. It allows for graphical creation of tables, views, and users. The application also allows for the creation of automatic backups, and has the ability to show the health of the table structure and database. MySQL Query Browser is a GUI for running queries, checking views, and editing data.

2.3.5 PHP 5.0.4

PHP, the PHP Hypertext Processor, is a server module and scripting language that allows server side dynamic websites to be created with a C-like syntax. It supports direct communication with MySQL and runs as a module for Apache.

2.3.5.1 Sessions

PHP supports the handling of sessions, which allows information about each user's individual connection to the application to be contained on the server. Upon logging in to the Canadian Blue Minus application, the session is created and filled with the user's credentials. Each restricted page checks to see if the user is logged in and has the proper credentials before displaying its contents; if not, the user is sent to the login page. When the user logs out, the session is destroyed, preventing her from accessing restricted pages.

2.3.5.2 Includes

PHP allows some degree of code reusability through the use of the include() function. Instead of entering the code to connect to the database on every page of the application, one can instead make a single page that establishes the connection and include it on all the pages that need to connect to the database. Similarly, a page including constants that will be used throughout the application can be created and included.

3 Evaluation of the Prototype

The Canadian Blue Minus application served its purpose of helping the research group gain experience in large scale software engineering projects. Canadian Blue Minus was complete in terms of the specifications presented to the research group by the Central Mississippi Chapter of the American Red Cross. After its completion and evaluation by the research team, the following issues were deemed necessary to be addressed in the design and implementation of iCare.

3.1 Reusable Modules

In order to make the system more easily modified, objects within the system should be made as modular as possible. This would allow for errors within the system to be diagnosed to a single module or a group of modules. Those modules could be upgraded on another identical system with out interrupting the service of the entire system of an organization. Once the modules were finished and tested they could be easily copied and placed into the organization's system with little to no interruption of service. Unfortunately, PHP does not allow for this functionality. To create the system in modules the next system, iCare, will be created using Java Technologies such as JSP, JSF, SERVLETS, and the Sun Java System Application Server.

3.2 Generic Design

Canadian Blue Minus was designed exclusively for the Central Mississippi Chapter of the American Red Cross. One of the goals set forth by the research group is to create a system that is generic enough for most non-profit organizations that deal with some form of volunteerism to be able to use it with little to no alterations. This goal would also allow for more organizations to use the system and promote more non-profit organizations to keep track of their volunteers. A generic design also lends itself to future work of having multiple nonprofit organizations using the same system to help each other. When one organization needs more volunteers, it would be able to ask individuals from another volunteer organization that has placed all the volunteers it can. This collaboration among nonprofits is the next step in volunteerism, allowing for the most volunteers to go to the place of highest need, even if the place is not considered related to the organization the volunteer signed up with.

3.3 Security Issues

Some security issues were not discovered until the completion of the prototype. The main issues that will be addressed in iCare are: password masking, user groups, and secure connections.

Like most online systems that require a password, the password field is masked so that no one can determine a password by watching the screen. One aspect of storing the passwords that did not occur to the research group is that the passwords are stored as VARCHAR, allowing anyone with sufficient knowledge of MySQL to sit at the server and use the MySQL Query Browser and the passwords for all users. Even though the research group believes that anyone who has direct access to the system would not consider this, it is possible and therefore should be addressed. This security issue will be resolved by using encryption within the code to store the passwords as 25 character alphanumeric strings that will be decrypted only by the application to authenticate a user. This allows the database to store the passwords securely so that not even the administrator would be able to retrieve a user’s password, but only reset it.

Canadian Blue Minus also offered only two user groups: Volunteer and Administrator. This poses a problem when a volunteer is needed to do a simple task like send out Thank You notes. The only way a volunteer would have access to this functionality is for an Administrator to grant her administrative privileges. The volunteer can now have access to the portion of the system that allows the creation of Thank You notes, but now can also access all parts of the system. To resolve this security issue iCare will have more user groups. The user groups will be broken down into three main categories: Volunteer, Employee, and Administrator. This will allow for more flexibility when assigning a user certain privileges and maintains the security of the Administrators group.

Using PHP to create sessions is the most secure way Canadian Blue Minus had to create a secure connection between the user and the system. Even though PHP sessions are overall reliable and secure, there are instances in which an attacker is able to compromise their security. Even though this act is unlikely to occur, it is the ethical standpoint of the research group to maintain the highest level of security possible. To resolve this security issue the research group turns again to the Java Technologies. The Java Technologies will allow for a secure connection using multiple steps. The user will login, creating a connection between the JSF page and the Servlet; the Servlet then creates another secured connection between itself and the database, making the connection reliable and as secure as possible. If an attacker wished to infiltrate the system, she would have to create both connections instead of just the one found in PHP.

3.4 Missing Functionality

The reason for some functionality being left out of the prototype was the time constraint placed on the research group by the Central Mississippi Chapter of the American Red Cross. Canadian Blue Minus was complete based on the specification by the Central Mississippi Chapter of the American Red Cross. The functionality that is presented missing is functionality that the research group wished to implement into Canadian Blue Minus, but, because it was not a system requirement and due to time restrictions, it could not be implemented. The functionality that is introduced in iCare is the ability to encapsulate all the parts of volunteerism. The definition of volunteerism that was associated with Canadian Blue Minus was all aspects that dealt with the management of volunteers. Volunteerism should really include all aspects of volunteering. This includes the volunteers, the organizations they volunteer with, the individuals they help, and where they help them.

Canadian Blue Minus managed volunteers independently from other aspects of the system. If a volunteer was disabled, they might accidentally be asked to volunteer at a location that was not handicap accessible. To resolve this blindness of the system, the management of volunteers within the iCare system is integrated with all other parts of the system when possible. This will allow iCare to be more effective in its volunteer management duties.

Organizations were limited to only containing members as volunteers. To add to the functionality of Organization they are also allowed to offer services in the iCare system. This way the nonprofit organization could now keep track of organizations that would be able to provide services that individual volunteers are not able to. Many organizations might list mass transit as a service they could provide while most individuals would not have the ability to offer a 20 person bus as a service they could perform. This allows the nonprofit organization using iCare to have another means of keeping track of skills and services they might need.

Canadian Blue Minus also was not concerned with the individuals that the organization helped. This is functionality that the research group feels necessary for any volunteer driven organization. The functionality to keep track of clients allows the organization to track clients as they move from one location to another and also gives them great statistical data such as their turn around and the number of individuals they helped within a month or year.

Much like organizations, Canadian Blue Minus was not concerned with locations. The ability to keep track of locations allows the system to do more work, which means less work for the employee or volunteer using the system. The locations contain information on what is available at these locations and what type of location it is. Shelters are a type of location that has a total number of spots that they can hold and other amenities that are available. Using the iCare application an employee or volunteer could route clients to the nearest shelter that meet the needs of the clients. This functionality saves time on the part of the employee or volunteer and also potentially the lives of the clients depending on the severity of their need.

4 iCare

After the completion of Canadian Blue Minus, a volunteer management system for the Central Mississippi Chapter of the American Red Cross, it was decided to address the shortcomings of the system while expanding its scope. The new system, iCare, takes volunteer management to the next level through the addition of the client, employee, location, and organization modules. iCare also allows for better volunteer management through integration with the new modules.

The client module adds versatility to iCare that Canadian Blue Minus desperately needed—the ability to keep track of clients, individuals who are currently being helped by the organization. iCare keeps track of clients in several ways. First, it allows tracking of clients as they move from shelter to shelter; this allows families to find missing or displaced members who have been located by an affiliate of the organization. Second, the information gained helps the organization understand their client turn around and maybe what they can do to improve it.

The employee module was designed to help differentiate between people who volunteer for the organization and people who are actually hired by the organization. Though these people may perform similar roles for the organization, volunteers must be handled differently due to their transient nature.

Many organizations have multiple locations or are affiliated with organizations that have multiple locations. The location module seeks to allow the two campuses to use the same system while maintaining separate identities. For example, an organization might have a soup kitchen at one location, a computer center at another, and a shelter at a third. All locations are overseen by the same organization, but each one offers individual services that must be considered.

The organizations module furthers the role of organizations to not only supplying affiliated volunteers but also supplying goods and services. Willing organizations can input the services they offer to be searched by the organization as needs arise.

4.1 Sparta - Web Application Server

Sparta was the most powerful city-state in ancient Greece. Unlike Athens, Sparta was built on strength and force, as a good web server should be.

4.1.1 Hardware

Sparta is a used Dell PIII 600MHz Dimension. It originally had 128MB RAM. The upgrade to 384MB RAM posed two problems. The FlashBIOS had to be upgraded to support more memory, and the upgrade failed several times. The floppy drive was then replaced, which allowed the FlashBIOS upgrade to finally be installed.

Sparta contains two hard drives: one fifteen (15) GB and one twenty (20) GB drive. For Linux to treat them as one thirty-five (35) GB drive, Logical Volume Management was setup to create a partition that spanned the two drives.

4.1.2 Network

There are two 10Mbps lines run to SHH 309, where the servers and development machines are located. These lines are flaky at best, however; they only work with one computer running Windows, even when shared through a 5 x 100Mbps switch. Several days were spent diagnosing this problem, which set development back quite a bit. The problem was finally resolved by running new line from SHH 308 under the SHH 309 door to connect to a 100Mbps switch in another room.

4.1.3 Operating Systems

Two operating systems, Solaris and Fedora, were tested on Sparta. Fedora, being the easier to configure and use, prevailed.

4.1.3.1 Sun Solaris 10

Solaris 10 is a Unix variant developed by Sun Microsystems. It has many things going for it; it is free, and, being a Sun product, it supports Java out of the box.

Solaris, like any operating system, also has many problems. Many tout its unparalleled level of security, but in this instance that ended up being unfavorable. There were far too many permissions to feasibly learn and then set within the time frame of the project.

To compound the matter, Solaris 10 suffers from a lack of documentation. Seemingly all the documentation available for administration assumes a high degree of prior knowledge. Without enough experience using Solaris, the configuration became overwhelming and troubleshooting errors nearly impossible.

4.1.3.2 Fedora Core 4

Fedora Core 4, as mentioned earlier, is a product of RedHat and the open source community, so a great deal of documentation is available. RedHat treats Fedora as a test bed for new programs and configurations, so things that work on other distributions might not necessarily work, at least not in the same way, on Fedora. For this reason, one should make sure to read documentation pertaining to Fedora when possible.

4.1.3.2.1 Network

To make Sparta’s hostname “sparta.cs.millsaps.edu” so that it could be seen across the network without having to know the current dynamic IP address, the following instances of the hostname had to be configured in Fedora Core 4:

- /etc/hosts
- /etc/sysconfig/network
- /etc/sysconfig/networking/devices

- /etc/sysconfig/networking/profiles/default/hosts
- /etc/sysconfig/networking/profiles/default/ifcfg-eth0
- system-config-network (GUI Network Configuration Utility)

Finding all these files took literally searching through every folder that might have anything to do with system configurations, which set back development a bit. The WINS server for the network also had to be configured to allow the machine to be seen easily by Windows clients.

4.1.3.2.2 Users

To keep from accidentally making changes to the system using the supervisor account, different groups with different privileges were created. A web administrators group was given authority to set web server configurations, while the developers group had no such power. Both were given read, write, and execute privileges in the directory where the web application was kept.

4.1.3.2.3 TTY Over Null Modem Cable

To allow remote administration of Sparta by a Windows client without opening up network vulnerabilities, an RS-232 DB9 null modem cable was built. Adding a line to "/etc/inittab" and "/etc/securetty" enabled Sparta to use mingetty to answer requests on /dev/TTYS0 (COM 1 under Windows). This lets a Windows client to open a virtual terminal from Sparta using a program such as HyperTerminal and Sparta credentials. An added bonus is that text can be directly copied and pasted to and from the terminal and the development machine.

The line to add to /etc/inittab is:

```
s0:2345:respawn:/sbin/agetty -L -f /etc/issueserial 38400 ttyS0 vt100
```

4.1.3.2.4 SELinux

SELinux is a sort of firewall that works along with iptables and ipchains to lock down ports on a Fedora host, providing a great deal of security. It caused a great deal of frustration and set back development by several days. Using the graphical configuration tool, the ports for Samba and the web server had been opened to traffic. After days of tinkering with settings and configuring other services trying to determine why these ports could still not be seen, it was found that a setting in "/etc/selinux/config" had to be changed from enforcing to permissive to allow traffic through the ports, despite the setting already having been changed in the graphical utility.

4.1.4 Software

4.1.4.1 Samba Server

Because of the hang-ups with SELinux, a great deal of time was spent trying to configure Samba for Sparta. This included the installation of S.W.A.T., the Samba Web Administration Tool. The final configuration for the share in the web applications folder used for deployment is as follows:

```
[Tomcat]
    comment = Tomcat
    path = /usr/java/tomcat/webapps
    valid users = <user1>, <user2>, <user3>
    admin users = <user1>, <user2>, <user3>
    read only = No
    create mask = 0777
    force create mode = 0777
    directory mask = 0777
    force directory mode = 0777
    inherit permissions = Yes
```

4.1.4.2 Java 5

Since Fedora Core 4 is used by RedHat to experiment with new settings, some libraries are not completely compatible with the standard ones. Installation of the J2EE 1.4 SDK and J2SE 5, both of which are required to run the application servers, required the installation of the libstdc++ and libstdc++-dev packages.

4.1.4.3 CVS 1.11.21

The Concurrent Versioning System (CVS) server maintains a repository of files that can be accessed simultaneously by several users. The server keeps track of every version of every file along with comments about new versions when new versions are committed to the repository. Several IDE's have CVS clients built in to allow collaborative project development.

4.1.5 Application Servers

The new breed of Java-based web applications, such as Servlets, Java Server Pages, and Java Server Faces, requires not only a web server, such as Apache, but also a J2EE compatible application server to handle the necessary connections, compile Java code into classes, and implement the Model View Controller (MVC) architecture for which these applications are famous.

4.1.5.1 Tomcat Server 5.5.12

Apache Tomcat is a J2EE compliant application server whose development is overseen by the Apache Group, the same organization that handles the Apache HTTP Server. Tomcat is open source, and its available documentation is a bit cryptic. As of this writing, the current version of Tomcat is 5.5.12, but the latest published documentation covers only 5.0, which is significantly different in its configuration. The following is a list of files that had to be modified to have Tomcat work properly on Sparta:

- /etc/profile - Sets environment variables
    - JAVA_HOME - location of J2RE
    - CATALINA_HOME - location of Tomcat
- server.xml – Settings for the Tomcat Server
    - Changed default listener from port 8004 to 8080
    - Tomcat can serve static and dynamic content
    - No need for Apache since content is mostly dynamic
- users.xml – User configuration for Tomcat Server
    - Grants access to Web Administrators Group
    - Declares whether or not Management and Administrator Plug-Ins are accessible
        - Management Plug-In
            - Allows web-based deployment of applications
            - Accessible only by Web Administrators group
        - Administrator Plug-In
            - Allows web-based server configuration for
                - Users
                - Data sources
            - Accessible only by Web Administrators group
- MySQL Connector/J
    - Java-based Level 4 driver
    - Allows Java Applications to connect directly to MySQL servers

Tomcat is a very capable application server, but deployment of web applications requires too much user intervention to make rapid development and testing within a short time frame unfeasible. Tomcat also has some problems handling applications developed using Java Server Faces and makes configuration for connecting to MySQL databases a bit tricky.

4.1.5.2 Sun Java System Application Server Platform Edition 8.2

Sun Java System Application Server, being a product by Sun Microsystems, is designed and touted to work seamlessly with Java-based web applications. It supports multiple virtual domains out of the box and has a built-in Java-based web interface to allow a great deal of modification to the server settings and easy deployment of applications. Under Fedora Core 4, it requires the proper installation of the libstdc++ and libstdc++-dev packages prior to its installation.

4.2 Athens – Database Server

Athens was the center of knowledge and learning during the times of Greek Goddess Athena. The MySQL Server is named Athens in this spirit.

4.2.1 Building the Server

Athens began as an ASUS motherboard with a Pentium III 500MHz processor and 512MB RAM. It was then placed into an old Gateway case that already contained a floppy drive and a 200W power supply. From that point Athens was upgraded with another 1GB of ram. Athens then received a 32MB graphics card, a 10/100Mbps Ethernet card, a SoundBlaster Live sound card, and a generic 24x CD Drive. Next Athens received its physical storage: a 20GB, 40GB, and a 60GB hard drive. The 20GB became the primary master, the 40GB the primary slave, and the 60GB the secondary slave. The Gateway case did not have the right setup for the switch that came with the ASUS motherboard. Athens finally received a power switch by breaking an old video card and using the metal piece that has the video connector screwed into it to mount the on/off switch and the indicator light. Athens was then booted to inspect the hardware configuration. The BIOS only showed 512MB of ram with DIMM 3 and DIMM4 being empty. Athens needed to have its BIOS flashed and updated. The ASUS website was searched first, where it was discovered that it was no longer supported by the company. Third party driver sites were searched next to find a BIOS from the year 2001 which was presumed to be better then the 1998 release Athens currently had. After flashing the

BIOS, Athens' problem was not resolved, but it was a little better. With the newer version of the BIOS Athens can now see 1GB of its 1.5GB. Another unforeseen problem was the extreme temperatures Athens could possible experience since its components were most likely not designed with this purpose in mind. To resolve this issue, two fans were created by stripping two power-supplies of their fans and their small three pin power connectors and securing the connectors to the fans by electrical tape. One fan was installed in front and the other in the back to increase air flow.

4.2.2 Installing the Operating System

Athens was originally designed to run Fedora Core 4 with MySQL 5.0. Fedora Core 4 was chosen because it is a free, highly stable operating system. MySQL 5.0 is the newest stable release and has new improved features such as data warehousing and extended InnoDB support. Fedora Core 4 comes with MySQL 4, which can easily be installed with the operating system. The first attempt installing MySQL 5.0 was a disappointing failure. After researching the error messages the findings showed that if MySQL 4 is not installed during the installation phase, then Fedora would not install packages that are needed to run MySQL. Athens was then reformatted and during installation all MySQL packages except the actual program MySQL 4 were installed. After following the directions found on multiple websites to install all MySQL components, MySQL still would not work on Athens. After looking back through the tutorials, a section on upgrading MySQL 4 to MySQL 5.01 appeared. With that idea as a solution, Athens received yet another clean installation of Fedora, this time installing every package dealing with MySQL including MySQL 4. Then, very cautiously, using the directions to upgrade MySQL 4 to MySQL 5.01 from the MySQL website, Athens received MySQL 5.0. After three hours of patience, Athens was rebooted to finish the upgrade. After the reboot the MySQL service was still not running; a little frustrated, the research group tried a manually start of MySQL 5.0. After about three seconds, Athens received an error message dealing with the execution of MySQL 5.0. After these three attempts and with a deadline for having the server operational coming up, Athens received an installation of Windows XP with Service Pack 2.

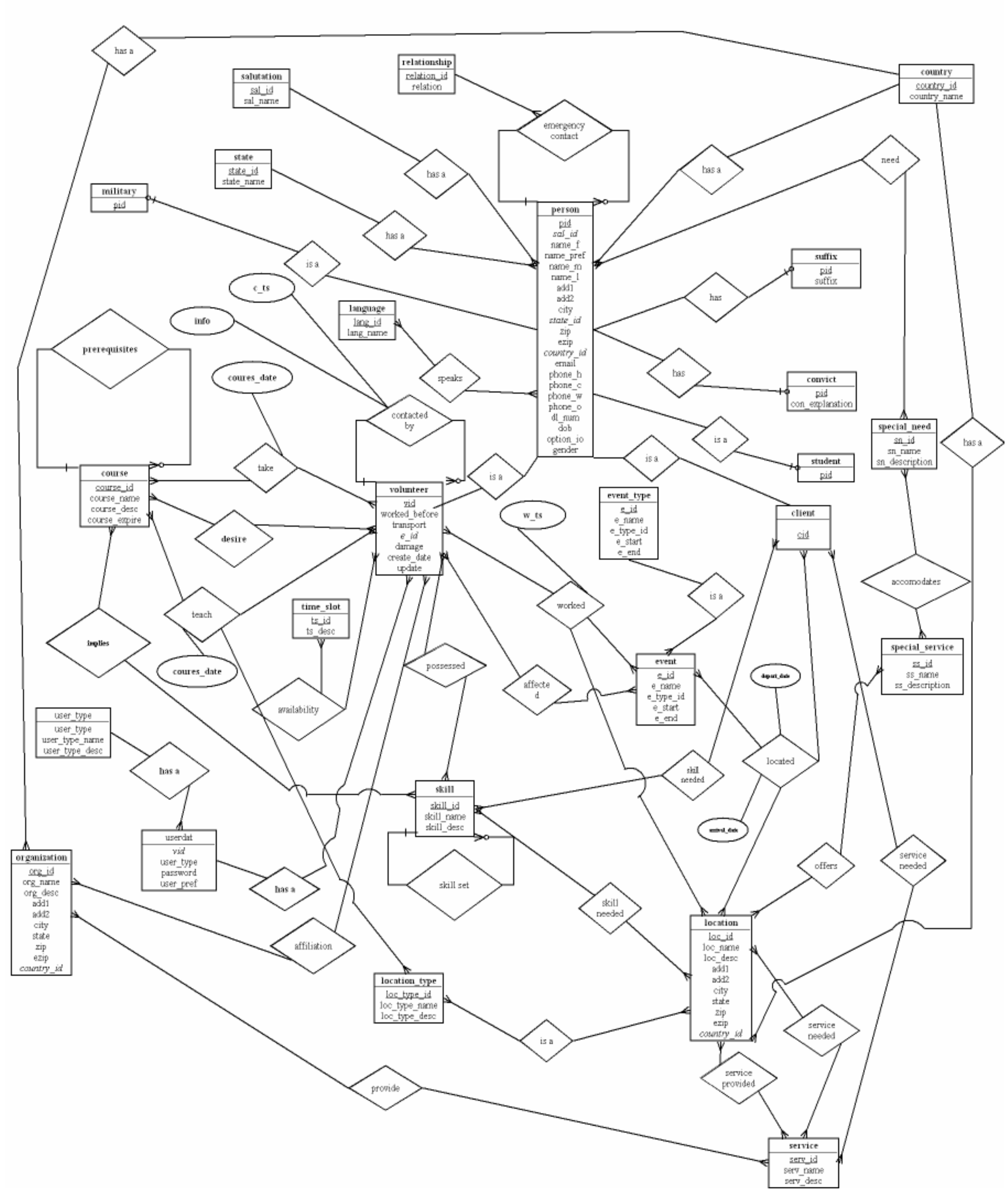
4.2.3 Installing and Configuring MySQL 5.0

After reading the tutorials found on the MySQL website, the installer package was downloaded to Athens, installed, and configured with no problems. The configuration chosen for Athens is quite unique. First Athens was configured to be a MySQL-MAX InnoDB Database Server. That is the MySQL way of saying that it is a MySQL server whose primary resources go to the application and that the default tables are InnoDB. Connection pooling was also set up to have a max of 5,000 simultaneous connections per user account. This was configured to such a high number to create actual MySQL users for the project and not just JSP and Servlet users; this will allow the system to have a much higher level of security and take full advantage of MySQL 5.0 and use USER VIEWS. The other configuration that is unique to Athens is the usage of the three hard drives. On the 20GB, where Windows XP is installed, MySQL 5.0 is also installed. The tables that are created are stored on the 40GB hard drive. This way, in case of a failure to the operating system, no tables have to be recovered, since they are on another drive entirely. The 60GB is the “pièce de résistance”; it contains multiple backup copies of the databases stored on the 40GB. Every morning at 5:00 a.m., the server shuts down the InnoDB process and creates a complete back up of all schemata on the server. As of this writing there are 56 backups of the database. This allows restoration from any day that the server has been stable and complete.

4.2.4 Creation and Optimization of the Database

The schema for Canadian Blue Minus only contained 25 tables, most of which were centered on the volunteer table. Additional tables were needed to overcome the limitations of Canadian Blue Minus to allow for the management of clients, employees, locations and organizations. The resulting iCare schema contains 55 tables.

iCare ER-Diagram



These tables were optimized to be as efficient as possible, especially in regards to load balancing between Sparta and Athens. The first step in optimization was choosing appropriate data types for the table fields. In general the smallest data type as possible was chosen to allow for quicker queries and for faster data transfer between Sparta and

Athens. This rule was only ignored if it was believed that it would be more beneficial for Sparta to receive data already formatted, making it easier for it to render the data. The best example of this occurrence is the way in which the database stores telephone numbers. The most beneficial way for Athens to store a telephone number is as a raw BIGINT (10) ex. 6019741000. To display this Sparta had to parse in the data and add the parenthesis and dashes. Since worse case Sparta would have to display four distinct phone numbers, it is more efficient to store the phone number as a VARCHAR (20). This allows the database to store the telephone number just as it would be displayed ex. (601) - 974 - 1000. With this form of load balancing Sparta could now display the data directly instead of taking the steps to parse the data. Another form of optimization of was the creation of indices for the InnoDB database structure. The InnoDB structure creates a new index with in a table whenever a column is declared as a foreign key. This not only creates and helps maintain referential integrity but also optimizes the SQL statements such as INSERT, UPDATE and SELECT [5].

The schema also contains Views that were created to decrease the number of prepared statements used with in the actual code. Since the upgrade from MySQL 4.1 to MySQL 5.0, the system gained the ability to add Functions, Procedures and Triggers into the schema as well.

A MySQL View, much like all views associated with database management systems, allows the creator, usually root or a high level administrator, to hide portions of the database or create portions from existing data for users. This ability was extremely helpful during the implementation, because it allowed the creation of the queries that were to be used even before the page that needed the information was created [5].

A MySQL Function is just like a function that would be created for a C++ or Java program. It takes in a single parameter or multiple parameters separated by a comma and returns a single value. The iCare schema has a Function that that takes in a username and password and returns the matching name. It seems like a simple enough SQL statement that should be just implemented within the code; but using a function allows making changes to the database and then making changes to the single function, instead of going through the code and rewriting the queries over and over again. The MySQL code to request data from a function getName() that takes in the username and password: CALL getName(‘username’, ‘password’); [6].

A MySQL Procedure is much like a Function, but instead of a single return value, it returns a temporary table. This table is created by a single query or multiple queries and allows the users to name the column heads that will be produced and their order. A Procedure could be compared to a view that takes in a single or multiple parameter variables. This is great for the generation of lists or other massive amounts of data that will be used frequently. One example is the data displayed when a user goes to check her information. The Procedure called getInfo() takes in the username and password of a user and then returns a table with all the information needed by the user info page. The MySQL code to request data from a procedure getInfo() that takes in a username and a password: SELECT getInfo(‘username’, ‘password’); [6].

A MySQL Trigger allows database systems to become event driven. A Trigger is set to occur when ever a table is altered or selected in anyway. The Trigger can also be used to call Procedures and Functions as well. iCare has a Trigger to update the table called data_base, that contains some of the table names and a timestamp. Whenever the tables are altered, the Trigger updates the table data_base with the current timestamp for that table. This allows the system to accurately estimate the completeness of a query based on the time the query was called and the last time the table had been altered [6].

4.3 Development Machines

4.3.1 Hardware

The recommended minimum requirements for working with Java applications are as follows:

- 2.4 GHz Processor
- 1.0 GB RAM
- 100 GB Hard Drive for each operating system
- 64 MB Video Card (if possible, have two)
- 17” Monitor capable of 1280x1024 resolution (if possible, have 2)
- 10/100 Mbps network connection

During the height of development, the motherboard of one of the development machines began to have serious problems. This appeared first as Windows XP problems, making it impossible to run the IDE’s necessary for development. As the problem worsened, the computer refused to boot Windows XP at all, and finally stopped showing output at all. Taiwanese tech support suggested rebooting the machine, at which point the board, processor, and RAM were replaced. As developing without a development machine can be difficult, this led to several days of setback.

4.3.2 Software

4.3.2.1 Operating Systems

4.3.2.1.1 Windows XP Professional SP2

Windows requires the least amount of work to setup and install applications, and it is the operating system with which the research team had the most experience. Windows also allows for easy configuration of dual monitors, which makes side by side development and testing possible. It is not always amicable to heavy Java usage, however, and requires frequent rebooting during development. All the IDE’s were Windows compatible.

4.3.2.1.2 Fedora Core 4

Fedora Core 4 is very sturdy, but updates are pushed to it continuously, making configuration difficult to maintain. There are currently only a few display adapters that allow dual monitors under Fedora, none of which were available for this project. Some of the IDE's ran under Linux, but others were for Windows and Mac OS only.

4.3.2.2 Integrated Development Environments (IDEs)

Most IDE's use syntax highlighting to aid in coding, and some even show the result of the code as it is written visually. Many of them also allow components to be dragged and dropped from a palette onto the workspace to make development fast and easy.

4.3.2.2.1 Borland JBuilder 2005

JBuilder is a good tool for creating simple Java applications, but it does not hold up well under strenuous loads. The free version, JBuilder Foundation, does not allow the creation of web-based applications.

4.3.2.2.2 NetBeans 4.1

NetBeans is a product of Sun Microsystems that is very similar to NetBeans, except that it is completely free and does allow the creation of Java web-based applications. It does not, however, allow the creation of Java Server Faces applications.

4.3.2.2.3 Eclipse 3.1.2

Eclipse is an open source IDE under the guidance of the Eclipse Foundation that uses plugins to expand its features to different languages, such as C/C++ and Java. There are plugins to develop Java web-based applications, but they are currently under development as well. There are no plugins to create Java Server Faces applications. Eclipse does have built-in CVS support to make collaboration very easy.

4.3.2.2.4 Macromedia Dreamweaver MX 2004 / 8

Dreamweaver 8 is the newest version of Dreamweaver available. It supports development in several languages, including HTML, XHTML, PHP, and others. It is the IDE used to develop the Canadian Blue Minus application. Version 8 is the first version to not support Java web-based application development. Dreamweaver MX 2004 does support the creation of Java Server Pages, but just barely, and it does not support Java Server Faces.

4.3.2.2.5 Sun Java Studio Creator 2

Java Studio Creator 2 is currently the only IDE that supports the development of Java Server Faces applications. It ended up being the IDE used for development of the iCare application.

4.3.2.2.6 Sun Java Studio Enterprise

Java Studio Enterprise has the same look and feel of Java Studio Creator 2, with the addition of a collaboration suite but with the subtraction of Java Server Faces development.

4.4 Java Server Faces

Java Server Faces is the newest edition to the list of Java Web Technologies designed by Sun Microsystems Incorporated. It was said that Java Server Faces was going to put a friendly face in front of the web application [7]. The same individuals who wrote the most current Java Server Faces book are quoted as saying “programming with servlet and Java Server Pages (JSP), we found to be rather unintuitive and tedious.” If nothing else the words spoken by the authors of a Sun Microsystems Press book carry more weight then the words of the research group.

4.4.1 Proposed Final Draft

The Proposed Final Draft 2 of the specification and its new implementation of Java Server Faces were started on February 15, 2006 [8]. As of this writing there is not an estimated time of completion of The Proposed Final Draft 2. There is also not a completion date for The Proposed Final Draft which was started August 25, 2005.

4.4.2 Compiling the Pages

One of the many shortcomings of Java Server Faces is the fact that each Java Server Faces page actually contains at least three files. These files include SomePage.java, SomePage.jsp, and SomePage.jsf. To render all three files as one page, Java Server Faces has to be compiled the first time it is displayed. Even a page that had no graphics, videos, sound, or any other multimedia could take up to a full minute to load the first time. This shortcoming was overshadowed by the fact that upon deployment to the Java Application Sever there was no way for the application server to compile them. Since all pages have to be recompiled after the deployment process, any application that is created with it would have to be navigated by hand to ensure that all pages compiled correctly. Another downside is the display time to users who use a 56K or less connection. If it takes a full minute to compile and display on a page on a 100MB intranet, it will take many times longer over a dialup connection.

4.4.3 Navigation

The navigation between Java Server Faces is all stored in a file called navigation.xml. Unlike HTML, PHP, or other server side web languages, navigation is in XML format where links and buttons return a meaningful string name instead of a direct link. That meaningful string was then described in the navigation.xml as an address for a button, tab of link to follow. This aspect of Java Server Faces was confusing at best at first encounter. Each page has its own navigation-rule tag that contains a select case or a navigation-case for each page. The second tag, called the from-view-id, lets the navigation.xml know that the rules that are going to be described only relate to this one page. The next set of tags describes each case; this tag is called the navigation-case. The two parts of it include the from-outcome tag which is the string that should be returned to activate its case. The second part of the navigation-case tags are the to-view-id which is the path that the link will follow. After closing all tags navigation would be possible. Below is an excerpt from the navigation.xml dealing with the page login.jsp and the two links that were on the page. When the string “logged_in” was returned the user was sent to user_Main.jsp, when the string “not_logged_in” was returned the user would go back to the login.jsp.

```
<navigation-rule>
        <from-view-id>/login.jsp</from-view-id>
                <navigation-case>
                        <from-outcome>logged_in</from-outcome>
                        <to-view-id>/user_Main.jsp</to-view-id>
                </navigation-case>
                <navigation-case>
                        <from-outcome>not_logged_in</from-outcome>
                        <to-view-id>/login.jsp</to-view-id>
                </navigation-case>
</navigation-rule>
```

The way that Java Server Faces deals with navigation is very cumbersome, especially to any individual who is not familiar with XML.

4.4.4 Layout Issues

Java Server Faces is often described as a web language that is concerned with the appearance and usability of the applications made with it. With that in mind one would think that Java Server Faces would be easy to layout and display through the internet the way that they appear in development; unfortunately, that is not the case.

The most difficult layout centered on the Java Server Faces tab sets. Once a new tab set is created it comes with a layout panel; the tricky thing is that no layout can be accomplished using this panel alone. To place a bare element in there, such as a text box or a label, the element will be positioned at the top left hand corner of the tab set. To work around this each element or each group of complementary elements must be placed within a grid panel. That grid panel may then be placed in the tab set. All other grid panels added to the tab set will appear in vertical flow. This attribute becomes nerve racking when the developer used to free form placement of objects, in which one merely places an object where it needs to go and it stays.

Another layout issue dealing with Java Server Faces is its inability to show relative positioning using CSS. Java Server Faces handles absolute positioning as long as an element is not in a container such as layout panel or grid panel, so logically it should handle relative positioning. This inability creates a lot of wasted time when testing layout positions. Once an element or group of elements has been laid out the files have to be deployed to the actual Sun Java Application Server or the Sun Java Studio Application Server Platform Edition, which takes a couple of minutes, before the changes could be viewed. For a web language to boast that it is concerned with the appearance and usability of applications then its layout or its claim should be reevaluated.

4.4.5 Resolving Errors

Java Server Faces has an interesting way of dealing with errors depending on the number that are found within a file. If a file contains only one error, then there is some workaround to allow the file to be complied and displayed properly. This unfortunately happens often when unfamiliar with Java Server Faces. If multiple errors occur, then bulletin boards and websites all give the same simple solution when dealing with Java Server Faces: delete everything and start form scratch. Starting from scratch involves using Windows Explorer and deleting any remnants of the file or files associated with the project. This gives the creator of the files much needed practice in creating Java Server Faces and also gives them negative reinforcement to do not it again.

4.5 Java Studio Creator 2

Creator is a fairly new Java Integrated Development Environment built with NetBeans 4.1 that aids in building layered Java web applications. Java Studio Creator 2 is needed when creating Java Server Faces since it is the only application that actually creates Java Server Faces.

4.5.1 CVS

The Java Studio Creator 2 feature that seemed most useful is its ability to allow for collaboration using Concurrent Versions System (CVS). Without the ability to use CVS there is no way for more than one person to work on the project at any given time. The ability to have multiple coders at any given time is essential due to the deadline placed upon the research group.

4.5.1.1 Add

Once a file was stable or deemed necessary to be added to the project then an individual would navigate and right click on the file in the Projects Pane and select CVS -> Add. This would add the file to the CVS repository after the first Commit.

4.5.1.2 Update

To insure that the files are concurrent with the files of other members of the research group an Update must occur. The easiest way to update all files is to right click on the project name in the Projects Pane and select CVS->Update. If an update did not take place then members of the research group would be working with different versions of the system which could lead to an enormous problem when an individual finally tried to Update the files within the project.

4.5.1.3 Commit

If a file was finished, or in a complete and stable mode after alterations, the programmer would then Update the file incase another individual had altered the file while the programmer was working on it. After the update the programmer would Commit the files, bringing the file to the newest version with in the CVS repository. This was accomplished by right clicking on the file found in the Projects Pane and selecting CVS->Commit.

4.5.1.4 Merge

After an Update, a file a programmer has been working on might have merge issues. To resolve these, the programmer would merge the two files using a side by side view of the code that was held on the CVS repository and the code they had created. The merge process was initialized by both Updating a file of by requesting a Merge by right clicking on a file located in the Projects Pane and selecting CVS-> Merge.

4.5.2 System Requirements

Java Studio Creator 2 has the heftiest system requirements of any IDE seen by the research group. The minimum system requirements for Java Studio Creator 2:

- CPU Intel Pentium 4 (or equivalent) at 1 Ghz processing speed
- RAM 1 GB
- Disk Space 650 MB (an additional 300 MB required for installation)
- Operating System
    - Windows 2000 Professional Edition (SP 4)
    - Windows XP Professional and Home Editions (SP 2)
    - Windows Server 2003
- Supported Browsers
    - Internet Explorer 5.5 Service Pack 2 (Windows 2000)
    - Internet Explorer 6 Service Pack 2 (Windows XP)

4.5.3 Handling of Deletion

One of the most recurring bugs with Java Studio Creator 2 is its inability to delete files correctly. As mentioned before, Java Studio Creator 2 edits at least three files per Java Server Face. Once you delete the main file SomeFile.jsp, the application does not always remove the other associated files. This causes errors when trying to compile or deploy the project. Then the siblings of the file that it says are missing or references to that file or set of files have to be found and corrected. Unfortunately for most users who are not experienced with Java Server Faces and the other Java Technologies used this error occurs all too often. This is another instance where Java Studio Creator 2 gives negative reinforcement when a mistake is made.

4.5.4 Documentation Not Yet Published

March 16, 2006 Sun released the Java Studio Creator Field Guide, Second Edition for review to anyone who had been using Java Studio Creator 2. Unfortunately its post came after four months of wrestling with the application and three weeks before the project was due. Java Studio Creator Field Guide, Second Edition is set to be published in the fall of 2006, and it is a recommended reading for anyone who would like to learn about and understand Java Server Faces before programming in the languages [9].

4.5.5 Updates Break Workarounds

With Java Studio Creator 2 being such a new application and being that the actual requirements of Java Server Faces have not been finalized and is still in draft format, there were often updates to Java Studio Creator 2. The updates to Java Studio Creator 2 were both a blessing and a curse. There was a chance that the updates would fix some of the issues that the research group had been facing, so the vote was to always accept the updates. Unfortunately, it was found that workarounds created by the research group to solve an issue would not function correctly or would not function at all after the updates were installed. When this occurred the files that were no longer working properly were completely deleted and started completely over to return the system to working order.

4.5.6 Random Bean Placement

Another issue with Java Studio Creator 2 was its ability to randomly place data gained from the database into different beans. All data gained form a database query is stored in a Java Bean; besides the Beans created by the research group there are three main Beans: Application Bean, Session Bean, and Request Bean. The main two beans used within the system are Application Beans and Session Beans. Application Beans allow for a query or data to be kept for any part of the application to use at any time. The Session Bean is used for data that is used per person. When dealing with the creation of registration of new users it was intended that elements that dealt with languages to be an application bean since every new user has the same languages to choose from. Unfortunately when creating the data source Java Studio Creator 2 automatically placed that data in the session bean, which would mean that the query would have to be run for every individual who registered, even though it was always the same information. The only workaround found was to either leave it alone or to copy the information from the bean it was placed in to the correct bean. The research group hopes that Sun Microsystems will release an update for this flaw in the next update session.

4.5.7 Sun Java Studio Application Server Platform Edition

Built into Java Studio Creator 2 is the Sun Java Studio Application Server Platform Edition. This an application server that runs on a development machine to test projects with out deploying them to the Java Application Server. This component of Java Studio Creator 2 lacks a lot of usability. Running both Java Studio Creator 2 and the Sun Java Studio Application Server Platform Edition is a complete system drain that has the ability to bring even the above average desktop computers to a grinding halt. The Sun Java Studio Application Server Platform Edition also does not allow for the deployment of one page. A programmer must deploy the entire project taking close to five minutes just to see if one page is working correctly. Sun Java Studio Application Server Platform Edition is another product that needs to be improved before the next version is released.

4.6 Sun Java System Application Server 8.2

Java System Application Server is itself a Java application that serves up other Java applications. As such, there is quite a bit of overhead to use it. It, as with most other Sun Microsystems products, looks beautiful at first, until it is actually used.

4.6.1 Documentation

Because this is a fledgling technology, the documentation for Java System Application Server follows in the footsteps of Sun Microsystems product documentation before it: minimal, viscous, and filled with errors. Any settings that were eventually set correctly did not arise from deciphering Sun documentation, but rather from applying things learned from Tomcat documentation and searching online.

4.6.2 Connection to MySQL

Connecting to MySQL from a Java Server Faces application can be accomplished as in PHP by explicitly establishing a new connection on every page, but it is much more efficient to have the application server establish a connection pool and request a connection through the pool when necessary. Doing so requires the MySQL Connector/J from MySQL AB and a good deal of configuration for Java Application Server. Here again, following the Sun documentation did not work and caused days of setback. The solution was found by reading a MySQL document. The following settings are necessary:

- Establish a new connection pool using the datasource classname "com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource" and not "org.gjt.mm.mysql.jdbc2.optional.MysqlDataSource" as instructed to do by Sun
- Use the "javax.sql.ConnectionPoolDataSource" resource type, and not the "javax.sql.DataSource"
- Create a JDBC resource using the new connection pool instead of using the connection pool directly.

The documentation lists several different places to put the MySQL Connector/J jar file, none of which worked independently. The combination that worked for Sparta was:

- /opt/SUNWappserver/lib/mysql-connector-java-3.1.12-bin.jar
- /opt/SUNWappserver/domains/domain1/lib/mysql-connector-java-3.1.12-bin.jar
- /opt/SUNWappserver/domains/domain1/lib/ext/mysql-connector-java-3.1.12-bin.jar

4.6.3 Deployment

The documentation lists several ways to deploy an application, such as copying a war file to "/opt/SUNWappserver/domains/domain1/autodeploy". Sadly, the only method that ever worked was using Sun Java Studio Creator 2 feature that deploys to a remote application server. This effectively locks developers into using Sun Java Studio Creator 2, and since Creator will only deploy to Sun Java System Application Server, the lock is symmetric.

4.6.4 Bugs

Approximately one week after finalizing the development environment using Sun Java Application Server, Sun Java Studio Creator 2, and CVS, a bug in System Application Server surfaced, halting any future progress. Creator could no longer deploy to Sparta, prohibiting testing any future development. The administration console also began throwing an exception whenever configuration of the iCare application was attempted. The error, "com.sun.enterprise.admin.common.exception.MBeanConfigException: Component not registered", turned out to be a bug that Sun documentation listed as unable to reproduce and therefore not handled, but searching online found many unhappy developers who had run into the same situation.

5 Conclusion

With the delivery and installation of the Canadian Blue Minus server, the Central Mississippi Red Cross can begin the transition from paper-based forms to computerized forms stored in a database. Instead of information being available to only those with access to the filing cabinet, volunteers and administrators will be able to access information at any time from anywhere. No longer will the Red Cross have to spend countless volunteer hours sorting through illegible, out-of-date files to find help; now, they can simply click a few buttons and find exactly what they need.

Java Server Faces has been shown to be a technology with huge potential but also myriad flaws. As more developers adopt and critique it as this research group has, Sun Microsystems should be able to forge it into the robust technology it should be. Java Server Faces did present the opportunity to work with brand new and experimental technology, which is perhaps worth the angst endured while doing so.

MySQL has proven itself once again a powerful, stable ally. There were no problems encountered while designing and implementing the database. The only flaws related to MySQL were from trying to marry it to inferior products, for which MySQL cannot be blamed.

Overall, the project induced a great deal of education on the part of the research group regarding web and database technologies while producing a usable system for a worthy organization.

References

[1] Schwartz, Dr. Donald R., Adam T. Huffman, and Jonathan F. Spencer. "Service Learning, Software Engineering, and Hurricane Katrina - A Case Study". *Proceedings of WORLDCOMP/SERP '06*. 2006.

[2] Huffman, Adam T., Jonathan F. Spencer, and Dr. Donald R. Schwartz. "Canadian Blue Minus Demonstration". Millsaps College. 2005.

[3] Spencer, Jonathan F. and Adam T. Huffman. *Volunteer Management Users' Guide*. 2006.

[4] Huffman, Adam T. and Jonathan F. Spencer. "Volunteer Management: 'iCare'". *Tougaloo College Mississippi College 3rd Joint Undergraduate Research Symposium*. 2006.

[5] Ian Gilfllan. *Mastering MySQL 4.* Sybex Inc. 2003.

[6] *MySQL 5.0 Reference Manual*. MySQL AB. <*www.mysql.com/doc/* >. April 8, 2004. Revision 1767.

[7] Geary, David and Cay Horstmann. *Core JavaServer Faces*. Sun Microsystems Press. 2004.

[8] *Java Platform, Enterprise Edition (J2EE) JavaServer Faces Technology*. <http://java.sun.com/javaee/javaserverfaces/>. Accessed April 9, 2006.

[9] Anderson, Gail and Paul Anderson. *Java Studio Creator Field Guide*. Second edition. Sun Microsystems Press. Publication pending.

A Development References

Anderson, Gail and Paul Anderson. *Java Studio Creator Field Guide*. Second edition. Sun Microsystems Press. Publication pending.

Ball, Bill and Hoyt Duff. *Red Hat Fedora 2 Unleashed*. Sams Publishing. 2005.

Bergsten, Hans. *JavaServer Pages*. Third Edition. O'Reilly Media. 2004.

Blair, John D. *Samba: Integrating UNIX and Windows*. Specialized Systems Consultants. 1998.

Chopra, Vivek, Amit Bakore, Jon Eaves, Ben Galbraith, Sing Li, Chanoch Wiggers. *Professional Apache Tomcat 5*. Wiley Publishing. 2004.

Converse, Tim, Joyce Park, and Clark Morgan. *PHP5 and MySQL Bible*. Wiley Publishing. 2004.

Duffy, Kevin, Vikram Goyal, Ted Husted, Lance Lavandowska, Sathya Narayana Panduranga, Krishnaraj Perrumal, Joe Walnes, Richard Huss, and Meeraj Moidoo Kunnumpurath. *Professional JSP Site Design*. Wrox Press. 2001.

Geary, David and Cay Horstmann. *Core javaServer Faces*. Sun Microsystems Press. 2004.

Gilfillan, Ian. *Mastering MySQL 4*. Sybex. 2003.

Hall, Marty and Larry Brown. *Core Servlets and JavaServer Pages Volume 1: Core Technologies*. Second Edition. Sun Microsystems Press. 2004.

Krause, Jim. *Color Index*. HOW Design Books. 2002.

Krug, Steve. *Don't Make Me Think: A Common Sense Approach to Web Usability*. Second Edition. 2006.

Laurie, Ben and Peter Laurie. *Apache: The Definitive Guide*. O'Reilly. 1999.

Meyer, Eric A. Cascading Style Sheets 2.0 Programmer's Reference. The McGraw Hill Companies. 2001.

Meyer, Eric A. *CSS Pocket Reference*. O'Reilly Media. 2004.

MySQL Administrator's Guide. MySQL AB. 2005.

Shneiderman, Ben, and Catherine Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Fourth Edition. Pearson Education. 2005.

Weinschenk, Dr. Susan and Sarah C. Yeo. *Guidelines for Enterprise-Wide GUI Design*. John Wiley and Sons. 1995.

Welling, Luke, and Laura Thomson. *PHP and MySQL Web Development*. Third Edition. Sams Publishing. 2005.

B Retrospect

If the project were to be "done all over again," it is not possible to say that Java technologies would be completely avoided, even though their use caused many unforeseen problems and wasted an immeasurable amount of time. Java technologies have great potential, and would be great tools for developing similar applications if a final, stable release could be agreed upon and proper documentation published.

If the idea of using Java Server Faces had never entered the picture, however, an easier to use language, such as ASP or PHP, would have greatly increased the development rate and allowed for a much larger scope. The reason Java Server Faces was chosen, though, was so that the research group could have a chance to use and learn cutting edge technology instead of merely implementing a project in a hackneyed language that does not pose any real challenge or offer any new innovations.

C Future Work

As was the initial plan for this project, iCare deserves to have an interface designed to match its impressive database. This would allow non-profit organizations besides the Red Cross to use the application to handle most of their operations. As of this writing, it is unfeasible to suggest doing so using Java Server Faces, but this will hopefully change in the near future.

The research group has already presented a demo of Canadian Blue Minus to the Central Mississippi Red Cross. Now that a server has been built to host the application, it will soon be delivered and installed for actual use by the Red Cross.

At the Tougaloo College Mississippi College 3rd Undergraduate Research Symposium, the research group delivered a presentation explaining the need to and the technology required to transition the Red Cross from paper-based to computerized records systems.

Along with their advisor, the research group plans to present a paper regarding service learning and software engineering at WORLDCOMP ’06 in Las Vegas, NV, in June, 2006.

Because of their interest in database systems, MySQL in particular, the research group is investigating studying for and becoming MySQL certified. This combined with their interest in software engineering has led them to pursue graduate education in information technologies.