Fall 2005

Conference Web Site Redesign

Steven M. Pisarski
Regis University

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

Disclaimer

REGIS UNIVERSITY

SCHOOL FOR PROFESSIONAL STUDIES

MASTER OF SCIENCE
IN
COMPUTER INFORMATION TECHNOLOGY

Conference Web Site Redesign

PROFESSIONAL PROJECT

Steven M. Pisarski
December 2005

ACKNOWLEDGEMENTS

I would like to thank my wonderful wife Janice, son Tyler, and daughter Autumn for their support and understanding during my time at Regis University, as there had been many late nights for me and boring weekends for us all at the library. Many times during this entire process of pursuing my Master's degree, my wife's constant encouragement has pushed me to persevere. I would also like to thank her for proofreading the countless papers, including this one. Without her help, I may not be graduating with honors.

I would also like to thank Karen Worminghaus for the time she had taken with me during the analysis and design phases of this project. Without her explanations of the conference, I would not have been able to develop a useful application. In addition, she also helped quell the many unrealistic requests from the director of the organization, which would have both increased the application's scope, as well as placed me into other tasks that would have led to me losing focus on the real task at hand.

ABSTRACT

Sustainable Resources was a non-profit organization based out of Boulder, CO that, at the time of this project hosted yearly conferences to help find solutions to world poverty. Although, the previous web site had a professional appearance, it was not meeting their needs with regards to usability, extensibility, and maintainability, as they had been forced to rely heavily on a transient volunteer IT labor base. The aim of this project was to address their functional and maintenance problems. The application's front-end was built in adherence to the Model-View-Controller (MVC) design pattern and implemented using the *Jakarta Struts* framework and consisted of, a JSP interface. *Hibernate* was used as the database persistence layer and the entire data model was completely replaced, as certain data fields were being duplicated in the former system. While giving Sustainable Resources the same functionality they had previously enjoyed, the new application enabled the site's administrators to effectively maintain parts of the application without IT support and supplied them with the necessary documentation to assist future developers with upgrades.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION/EXECUTIVE SUMMARY

A. Problem/Thesis Statement

Sustainable Resources was a non-profit organization based out of Boulder, CO that hosted an annual conference at the University of Colorado at Boulder to help find solutions to world poverty. The purpose of this project was to deliver an application that served not only “static” and dynamic Web content, but also provided the mechanisms necessary to manage user accounts, finances, donations, multiple conferences, and different participation types including: several types of tickets, presenters and their presentation(s), exhibitors and their spaces, sponsors, workshops, and volunteers.

When first contacting Sustainable Resources, they had expressed concerns regarding their current Web/ECommerce site. Although it had a professional appearance, they still relied heavily upon manual processes to handle the day-to-day activities. The secondary objective of this project was to provide a properly designed application that could be easily extended and maintained by nearly any Java web developer. This was probably the most important aspect to ensure the long-term success of this project. In response to this goal, the developer utilized a three-tier approach, which basically separates the presentation or user interface (UI) from the business logic. The business logic was further decoupled from the service tier by use of the Factory/Interface pattern. By taking this approach, future changes need only be made in a single part of the code thus reducing the total impacts. As long

as the interfaces to each entry point remained the same, one could be certain that the changes could be made in isolation.

B. Review of Existing Situation

After analyzing Sustainable Resources' current web solution, many problems were identified making it impossible to utilize nearly any of the existing code. It had been decided between the business owners and the student to rebuild their site from the ground up. The list below briefly summarizes many of the issues with which they had to endure:

- Inability of the administrator and business users to access or extract much of the data required for analysis and correspondence.
- No facility was available to manage any other participation types other than presentations.
- Inability to support retrieval the web content from inactive conferences due to the fact that the existing static HTML pages contained static links without relative paths making it impossible to simply place the old files into another directory.
- The data contained within their database was bound to the site and not a conference, therefore, when switching the site over to the subsequent conferences, all of the data would need to be removed by an IT resource. This act also disallowed the organization to maintain any historical records, as they did not have any mechanisms for retaining this data other than within spreadsheets.

- Their existing data model was not truly relational. As a number of data fields were duplicated into other tables, issues regarding consistency with their data became a growing problem for the organization.
- Navigating through Sustainable Resources site was sometimes a daunting task. Again, as they employed static HTML files that were being managed by a non-technical resource, the menu links were not always consistent from page-to-page, as each file would be impacted.
- Although the site contained a dynamic piece that retrieved presentations from a database, the original developer intertwined the presentation or Java Server Pages (JSP) with the data access, which made the extension of new functionality difficult to implement.
- To manage the sites content and financial capabilities, several disparate applications were required. By not encapsulating all of this logic into a single application, the costs associated with managing the web site was more than required with the new solution and the training of volunteers became very difficult.
- Lack of application documentation caused many maintenance and extensibility problems. As the original volunteer/developers were many times not available to make the necessary change(s), the transient volunteer programmers were having problems understanding the application.

- Sustainable Resources only had access to the current production code and regularly promoted untested changes. This aspect increased the risk factor of bringing down the entire site.

C. Project Goals – Relevance

Although ensuring that the application functioned as required, the ultimate aim was to achieve full customer satisfaction. The following attributes had been identified to be necessary for the delivery of a high-quality product.

1. **Stability**

   It was of primary importance that the application was stable. Sustainable Resources did not want to lose potential participation in their conference for which they spent the entire year preparing.

2. **Functionality**

   By utilizing an iterative methodology during the development lifecycle, it was the author's hope that any problems with the expected functionality could be caught early in the lifecycle. If this aspect was not meet, this project could have failed.

3. **Ease of Navigation**

   An application that fulfilled all of the business and technical requirements but was difficult to use could still be considered a failure. As a general rule, one should easily navigate to any desired page within three to four clicks, as the average person will generally lose interest if they cannot quickly find what they are looking for. Therefore, the developer

had put a strong emphasis on making the UI experience both visually and personally pleasurable.

4. **Maintainability**

As non-profit organizations generally have problems retaining IT resources, especially ones versed in Java and J2EE, considerations for the maintenance phase needed to be taken into account. Therefore, it was imperative that each design aspect be carefully considered. However, even the best-designed but poorly documented application could still have maintainability issues. Therefore, great pains were taken to find an UML tool. The expressive diagrams produced during the design phase was the first step in creating a deliverable that could be handed off to future developers to assist with repairs and enhancements.

D. Issues and Barriers

1. **Financial**

This entire project was produced with a minimal budget. The student's employer in the form of purchasing the UML tool and reference books provided some funding, however, Sustainable Resources were solely responsible for the site's web hosting costs. The costs incurred by the student consisted of the purchase of a small Linux server to function as a test environment. This box was made available using the student's static IP address via his broadband connection.

2. **Time**

The original plan to roll out all user functionality was initially targeted for October 2004. This date was very important to Sustainable

Resources, as they would have liked to introduce it during their fall 2004 conference. The administrative maintenance and reporting functions were planned to follow shortly thereafter with a targeted completion date around December 2004. As the student underestimated the time necessary for analysis, development, and encountered inevitable scope creep, the application was not ready for system testing until August 2005.

3. Development Software

As it was planned to develop the application using the Rapid Application Development (RAD) Software Development Lifecycle (SDLC) methodology, the choice of the proper development tools was important to the success of this project. There were many products on the market, however, as the budget was small, finding a cost-effective, completely open source solution was not an easy task.

4. Maintenance Phase

Java technologies were once esoteric to many developers, however with the adoption of Java/J2EE technologies by many companies, the number of Java developers had continued to increase. Once learned, these technologies offered many benefits and sped development and helped streamline maintenance.

It was planned, however, for the student to maintain Sustainable Resources' site indefinitely. If the application was designed correctly, repairs and enhancements to the application should have taken too much of his time. In the future, it was expected that more willing volunteers

would understand this technology and could take over the maintenance of the application. It was planned to deliver tools, resources, and documents, which could allow for a swift transition.

5. Administrative Technical Knowledge

By their own admission, Sustainable Resources' staff was not computer literate other than using common types of applications such as: browsers, email, word processors, and spreadsheets. Karen Worminghaus, however, was found to be an efficient user and easy to train. Although the student did not try to overwhelm the customer with many of the technical details, basic knowledge of some high-level concepts pertaining to relational databases was transferred to ensure the administrative users understood their data and how it could be extended in the future. Certain database tools had been suggested to the organization so they could learn how to issue ad-hoc queries not supported by the application.

6. User Technical Knowledge

As this application was deployed on the Internet, anybody with enough knowledge to operate a web browser could access the site. It was of primary importance that it was designed in such a way that everybody could easily understand and navigate, therefore, its pages were all built using *Struts* templates called Tiles to help ensure basic web navigation standards were employed in a consistent manner.

7. Discontinuance of Conference

To the chagrin of the author, after contacting Sustainable Resources upon completion of the development phase, he was notified that the plans for future conferences had been placed on hold. Therefore, at the time of completion of this document, this application had not been deployed to a production web server. In addition, no testing resources have been allocated by the organization, thus, the 15 plus months of design and development activities had been in vain. This was especially disturbing to the student as the initial release was postponed due to the design and development efforts required to address the late requirements of supporting multiple conferences in multiple locations, content management, and support for archives in December 2004.

E. Scope and Limitations of Project

1. Initial Scope

Initially, Sustainable Resources requested that the application only manage a single conference, and they would continue maintaining the majority of the application's content. However, an evaluation was made on how they were handling the static HTML pages, as they wanted to have full-control of the site. In addition, as they were handling nearly all aspects of hosting their annual conference manually, the business manager of the non-profit defined the primary goal of the application to maintain member data and how they would participate in the conference.

2. Actual Scope

During both the design and development phases, there were many actual requirements that were being fleshed out. The resulting application morphed into much more than what was originally anticipated from the initial meetings. In addition to the scope outlined above, the application's functionality increased to include financial transactions, addition of several new participation types, a new methodology to maintain the sites content that included mechanisms to enforce ease of navigation, handling of multiple conferences instead of just one, and ability to retrieve content from past conferences.

3. Application Limitations

Although the business owners could get access to the applications database, they did not understand how to view the data. Even though it would have been nice to implement a Data Warehouse schema on top of the on-line transaction-processing (OLTP) system, as relational models are not always intuitive, the creation of an entire schema along with the scripts and database triggers such an approach would require was determined not to be feasible. Therefore, the application could have benefited greatly to some sort of reporting mechanism.

Even though the application employed a "factory/interface" pattern to decouple the business layer from the data access layer, the business delegates contained simple messages to static methods required of the Object Relational Mapping (ORM) tool, *Hibernate*, utilized for database persistence for multi-threading purposes were placed into this tier. This

design flaw was due to the developer's lack of knowledge of the tool. Although the application functions well, impacts, albeit minor, to the business layer would occur if a decision would be made in the future to change the persistence mechanism to something like Container Managed Persistence (CMP) Enterprise Java Beans (EJB).

Due to time constraints, functions primarily to do with searches were omitted from the initial release. User search functionality was limited to last name only and participations by type and status. Additionally, due to non-participation on the part of the organization, the system-generated emails were only populated with persistent data. As the developer never received any detailed requirements for these functions, static data was not employed.

As the organization had discontinued their annual conferences, the application had never been put through either system or production readiness testing, which was supposed to occur utilizing resources provided by Sustainable Resources; therefore, unexpected defects probably still exist. In addition, the credit card processing functionality provided had never been tested, as they never purchased the software required by their processing service, and, as these transactions required a secure transmission, the organization did not have a valid SSL certificate either.

SSL encryption of all form data transmissions was also supposed to be implemented as well during the system test phase. However, as the

organization lost interest in the project, the student had not gotten to this important feature.

Lastly, the developer never updated the scripted JUnit tests of the business and persistence layers, which were refactored after their initial development. As the requirements changed throughout the project, it was necessary to make some major changes to the data model, and the ones that took place while the user-interface (UI) was being developed, these changes were never updated to the scripted tests.

F. Definition of Terms

Please see the Glossary at the end of this document.

G. Summary

Sustainable Resources' web site primarily served up static data, which was maintained using a standard HTML editor. On the surface, the site had a professional appearance, however, the many revisions made by non-technical personnel had created inconsistencies in many of the standard menus making navigation difficult at time. Although the site had some dynamic aspects, the data source from which these pages received their data was not useful for any other function. As it was the goal to produce an application that could accept and display pertinent data about specific conferences, it had been determined to create an entirely new application that would meet the organization's needs in an effort to increase the productivity of the few employees and volunteers.

II. RESEARCH

A. Literature

1. *The Design of Sites* by Douglas K. Van Duyne, James A. Landay, and Jason I. Hong

Although the student did not read this text from cover-to-cover, he did however, gather many insights into the world of Web sites. The authors neatly organized the book to allow one to quickly find the concept being sought. The many layout patterns (not to be confused with code design patterns) were meant to be technology independent, but gave the reader good examples on how to structure a site that fosters efficient navigation. The single most important lesson gathered from this text is that all pages within a site should be accessible by no more than 4 clicks.

2. *Designing Flexible Object-Oriented Systems with UML* by Charles Richter

As many aspects of this application were designed prior to writing any code using UML and the student never had created a full set of diagrams for a single application, this text was an invaluable reference, as Richter provided excellent examples of many diagram types. The biggest knock on this book, however, was with its examples of Sequence Diagrams, as they did not seem to follow the notation standards supported in the chosen tool for this project. However, to Richter's defense, the book was written in 1999, and UML had been quickly evolving since that time.

3. *Apache Jakarta-Tomcat* by James Goodwill

This book was purchased so the developer could gather more information on the server of choice. He was expecting this material to

delve more deeply into Tomcat, however, the book merely glossed over Tomcat's more common features. It was the student's hopes that this text would have provided much more insight into the server's configuration and security model.

4. ***Hibernate in Action*** **by Christian Bauer and Gavin King**

This excellent tutorial on the currently hottest Object Relational Mapping tool was nearly all the developer required to quickly begin using this powerful tool. Being quite small for a book of this genre (< 400 pages), Bauer and King utilized each of the pages to its fullest and kept the reader engaged throughout. The authors began by offering valuable information about nearly all aspects of the concept "persistence". Overall, Bauer and King excelled in demystifying *Hibernate* and database persistence, which helped to drastically reduce the slope of the learning curve found whenever a new technology is pursued.

5. ***Struts Kick Start*** **by James Turner and Kevin Bedell**

The title of this book says it all. The student found the information contained within this text to be extremely informative and helpful. Besides giving one an excellent overview of the Jakarta Struts libraries, it offered excellent examples to begin developing functional dynamic web applications. Technical books generally come in two flavors. They either function as a tutorial or a reference. Turner and Bedell successfully authored a book that functioned in both manners.

However, where this book excelled in helping one write web applications using the popular Model View Controller (MVC) design pattern, it did not describe everything *Struts* offered. For instance, as the framework contained a robust library of custom JSP tags called a *Taglib*, it merely outlined some of its more common features, which could be said of all other aspects of Struts higher-level functionality, It did, however, educate the reader enough to allow them to extend their learning by other means.

6. ***More Servlets and JavaServer Pages*** **by Marty Hall**

This reference manual was designed to give the reader a good understanding of servlet containers and the standards around creating applications within these types of servers. The student, however, did not find this text to be overly helpful, as the application was build exclusively using the Struts framework that contains a single servlet, which controls the presentation layer.

7. ***JavaServer Pages*** **by Hans Bergsten**

Unlike Hall's book mentioned above, this reference guide was more useful regarding the development of JSP pages. As it focused upon JSPs only, Hall was able to hit on the most important aspects including the *Java Standard Tag Library* (JSTL). However, although he spent some time on *Struts*, he did not expand upon its tag library.

8. ***Cascading Style Sheets: The Definitive Guide*** **by Eric A. Meyer**

Cascading style sheets (CSS) are utilized by web browsers to help enforce a consistent look and feel throughout the site. This book was

purchased prior to the decision the Sustainable Resources' existing CSS would be leveraged. This decision was made to ensure that the new application would maintain the same look and feel as the old. Therefore, this reference manual was only utilized to gather a high-level understanding of how to access the existing style sheet's definitions.

9. ***HTML & XHTML: The Definitive Guide*** **by Chuck Musciano and Bill Kennedy**

Even though the application's front-end was developed exclusively with JSP, the output was ultimately HTML, therefore, a comprehensive knowledge of this language was gathered. Musciano and Kennedy had drawn up an excellent guide to all of the features offered by HTML and the developer found this guide extremely useful.

10. ***Java Cookbook*** **by Ian F. Darwin**

As the developer had limited experience developing Java applications as extensive as this, Darwin's reference guide was found to be helpful in implementing many of Java's useful features not covered in the coursework offered by Regis University.

B. Web Based Resources

Although an extensive library of reference material had been gathered during this project and the student's tenure at Regis, it had been found that there were many online resources available, which made research into common coding problems much easier to locate.

1. **Google**

   The search engine by Google was by far the most important research tool utilized during the development phase. As the developer did not always have the reference materials or desire to look up the answers provided in the texts listed in the previous section, the search engine was leveraged extensively to help locate solutions to most problems efficiently.

2. **On Java by O'Reilly**

   Hosted by one of the major vendors of computer technology books, this site contained a wealth of knowledge regarding nearly all J2EE technologies utilized in this project. By offering excellent observations and samples, the student found himself visiting this site often.

3. **Sun Microsystems**

   As Sun Microsystems originated the Java language, they had some of the most useful information. The developer primarily found the API documentation of most help. These "Javadoc" pages outlined all of the classes contained in the Java standard library. In addition, Sun's site contained many downloads, tutorials, and code samples for some of the most common problems encountered.

4. **Apache / Jakarta**

   The Apache Software Foundation was a community owning many open-source software projects, which has become highly respected amongst software professionals. Much of the software put forth by this organization had actually driven some of the standards, which many programmers take for granted.

Jakarta contained a subset of open-source projects consisting of Java solutions. The excellent software and documentation provided by this site enabled the developer to properly develop the application. The following three Jakarta projects had been utilized for this project.

a) Ant

Ant, a Java-based build tool, unlike other build tools such as *make*, utilized system dependent shell-based commands. Ant, however, relied upon an XML file, which could be utilized anywhere. The "targets" within a *build.xml* file defined all actions that needed to take place when building and deploying an application. Some of the most common operations Ant performs were: compilation, creating directories, copying files, packaging, and transferring files using the File Transport Protocol (FTP). Additionally, custom and third party targets could be utilized for performing more specialized functions.

b) Jakarta-Tomcat

Tomcat was servlet container adhering to the Java Servlet and JavaServer Pages specifications published by Sun Microsystems. The developer utilized this site to assist with the configuration of the application server.

c) Jakarta-Struts

Struts was developed in an effort to encourage development of applications use of the Model-View-Controller (MVC) design pattern. This product was designed to run in any servlet container and did not require the developer to write a single servlet. Instead, *Struts*

contained a single servlet that brokered out the calls to helper classes configured in an XML file. *Struts* also helped to foster good UI design when using Tiles, which made it easier to tie together multiple JSP files.

5. **Hibernate**

   The *Hibernate* web site was used extensively during development of the persistence layer. As this had been a relatively new product at the project's inception, very few reference materials were available. However, as this project site contained excellent extensive and comprehensive documentation, the developer found this toolset relatively easy to implement.

6. **Wikipedia**

   As acronyms were used almost as commonly as words, the English language appeared to be in some sort of evolutionary period. This online encyclopedia was a valuable resource for finding the meanings of these acronyms quickly.

C. Review of Existing Technical Solutions

It was required of all integrated development environments (IDE) evaluated prior to kicking off this project to either have built-in UML modeling features or have the ability to extend this capability via a third party plug-in. As there were many products on the market meeting these criteria, special attention was placed on ones employing round-trip development, which denotes a tight integration between diagrams and code.

As the *Ant* build tool had become the standard amongst Java build tools, the developer determined that the IDE needed to provide support for this tool as well.

1. Integrated Development Environments

a) Eclipse 3.0

This open source project originally developed by IBM had been long acknowledged to be an industry standard for all IDEs. Many third party plugins were also available for this tool that allowed its functionality to be extended. This extensible development platform compiled Java source while one coded allowing for syntax errors to be shown prior to building thus leading to greater productivity. Some of the other productivity tools embedded into this platform included automatic stubbing of class constructors and methods signatures, generation of getters and setters, wrapping statements into “try/catch” blocks, creation of main methods, automatic importing of packages of classes contained within the project’s classpath, and auto complete of variables, classes, and methods.

b) NetBeans

This free distribution contained many of the same features as mentioned above with the Eclipse product.

c) JBuilder

Developed by Borland, JBuilder was a robust development environment rivaling the abilities of Eclipse by allowing for third party products to be integrated into the software. However, as this was not

free software, it contained more built-in features not included in the base release of Eclipse such as: and XML editor, web service wizards, and integration with application servers.

2. UML Design Tools

As the developer was looking to increase efficiency, it was his desire to find a tool that could be plugged into the chosen IDE and offer round-trip development.

a) Rational XDE

Rational was one of the industry leaders and pioneers in the development of UML software. Their XDE and Rose products had raised the bar by which all other tools were judged. The distribution evaluated prior to the analysis phase was obtained through Regis University and had been built on top of the Eclipse 2.1 platform. After making some initial class diagrams, this distribution did not allow for round-trip development. The costs of commercial distributions of these products were prohibitive.

b) SDE by Visual Paradigm

Visual Paradigm was a young company that specialized in the development of productivity tools for IT professionals. The *Smart Development Environment* product was evaluated primarily for its UML diagramming supporting integration with code. It could also be plugged into most of the popular IDEs on the market. This product came in six distributions ranging in price from free to over $1000.00.

However, only the Professional and Enterprise editions allowed for round-trip development.

3. Struts/JSP Editors

As it had been the developer's goal to locate tools that would help speed development times, it was his hope to find a WYSIWYG JSP editor.

a) Dreamweaver MX

This product from Macromedia had long been a mainstay in web developers' toolkits as an HTML editor, and Dreamweaver MX was one of their first distributions supporting JSP. As Sustainable Resources had been using Macromedia products, it was only logical to look into this mature product. The primary drawback found by the developer was its support for custom tags was limited at best.

b) NitroX by M7

Initially released late 2004, this product appeared to contain everything the developer was looking for to help foster rapid development and prototyping. Although other products that claimed the ability to give developers the ability to develop JSP pages visually, *NitroX* was the only one that could accurately render each standard and custom tag and while fully supporting the Struts framework.

4. Servers

As a web site obviously required a server, careful consideration was taken prior to development to find the correct fit.

a) HTTP server by Apache

The widely used open source web server was designed to deliver static content over the Internet or an Intranet. It could also be leveraged to deliver secure content using the Secure Sockets Layer (SSL) protocol.

b) Tomcat

An open source Java based web server and servlet container used to serve up static and dynamic content in the form of HTML and *Java Server Pages* (JSP).

c) JBoss 3.2

JBoss, another open source product, was a full-fledged application server supporting the EJB 2.0 specification. For a free application server, it was surprising finding the number of large companies using this product. As JBoss was packaged with *Tomcat*, it could also serve as a, web server and servlet/JSP container.

d) PHP

This was yet another server designed to support dynamic page creation and utilized its own propriety programming language. This server was popular for the web sites of non-profit organizations.

e) .NET

Although Active Server Pages would have performed well for this project, it had been ruled out for the following reasons:

- The student wanted to ensure platform independence by using Java based products and technologies.

- The student's coursework had been focused primarily on Java, and he was unfamiliar with C#, ASP, and .NET.
- It was the student's goal to support the open source community and its standards.

5. Presentation Tier

The following solutions had been reviewed for the application's presentation tier based on the premise that it was web-based utilizing Java technologies.

a) Java Server Pages (JSP)

JSP allowed for HTML to be generated dynamically at runtime. A servlet container basically took the JSP code, generated a Java class and compiled it on the fly. Each of the following two solutions had been built on top of this technology.

b) Jakarta Struts

This product helped to facilitate the implementation of the *Model-View-Controller* (MVC) design pattern. This framework was designed to act more than a UI tool, as it made communication with the server more efficient by managing the contents of messages and each HTTP session.

c) Java Server Faces (JSF)

JSF was the latest presentation framework that had the possibility of replacing the more mature Struts. Although they both utilized a

single servlet to broker out requests and manage sessions, JSF used the concept of a page controller versus Struts' back controller.

6. Persistence Layer

As this application was heavily reliant upon persistent data, it was important to review several databases.

a) Oracle

When one thought about a database, Oracle was the first name that came into the minds of most. This highly scalable database would have to at least be considered for any project, due to its robust feature set and distributions for nearly every platform. Its major drawbacks for this project were primarily due to price and overall size.

b) DB2

One of Oracle's main competitors, this IBM product offered many of the same features of Oracle including size and price.

c) PostgreSQL

This free Structured Query Language (SQL) database, like each evaluated in this section, offered all of the required features such as foreign key constraints and indexed columns.

d) MySQL

Arguably the most popular free database, MySQL contained distributions for nearly every major platform offering arguably the best flexibility at the lowest cost. Being very mature, MySQL's web site had extensive documentation making implementation of this database quite easy.

7. Data Persistence and Object-Relational Mapping (ORM)

As it was required that this application leverage a database, the method of the basic create, update, delete (CRUD) operations needed to be defined either by writing JDBC queries, or by using an ORM tool.

a) SQL

Most database transactions take place by means of SQL, and many programs embed SQL queries and access the database using specialized application programmer interfaces (APIs), in this case, it would be Java Database Connectivity (JDBC).

b) Enterprise Data Beans (EJB)

This ORM standard is quickly losing favor amongst the IT community, however, they still hold a major share of the market and projects should not be evaluated without mentioning them. EJBs required that a J2EE application server such as JBoss be employed. Using the same types of interfaces as session and message driven beans, which are generally utilized for the encapsulation of business logic, **entity beans** are utilized as a persistence mechanism. The EJB 2.0 specification contains two different types:

Bean Managed Persistence (BMP)

These types of beans utilized the container's database pooling and issue SQL statements to the database by queries defined in a deployment descriptor.

Container Managed Persistence (CMP)

CMP beans defined the data base tables and their columns in a deployment descriptor which was used by the container to build the SQL commands with the assistance of the more generic EJB Query Language (EJBQL). The benefit of this approach was that one needed to only make changes in one place whenever changes to the database were required. In addition, as the syntax between databases could vary slightly, when changing database vendors, these same beans could be leveraged without changes.

c) Hibernate

This software was designed to make database persistence more highly abstracted and easy to employ while being lighter-weight than EJBs. By using a series of XML files that described a databases tables and columns, they could easily be manipulated by use of plain old java objects (POJOs). This framework reduced the amount of refactoring required when changes to the data model became necessary.

d) Java Data Objects (JDO)

Developed by Sun Microsystems, Java Data Objects appeared to have all of the makings to become a strong contender to Hibernate as a favorite ORM tool.

e) Expresso

This product encompassed the abilities of *Struts* and *Hibernate;* however, very little research was put into this product, as it was

decided early on to focus upon products that are being more widely used in industry.

8. Content Management

a) OpenCMS

This professional level open source website content management system was designed to create and manage complex websites without the user having any knowledge of HTML by use of an integrated WYSIWIG editor.

b) Macromedia Contribute

This tool was being used to manage Sustainable Resources' static HTML pages. It allowed for direct upload and download access to the server via FTP making it possible for administrators to deploy changes on the fly.

c) HardCore HTML Editor

This low-cost Javascript based in-the-browser HTML editor allows the user to upload images to the server for display and a wizard type approach to creation of hyperlinks when utilized in an application server. This editor also allows for direct import of *Microsoft Word* and *Excel* documents via a system's clipboard. It also supported custom styles when importing the site's CSS. An implementation such as this would require a more customized approach to CMS.

d) Editlet HTML Editor

Similar to HardCore but more costly.

e) pinEdit HTML Editor

See Editlet.

9. Reporting Solutions

Although reporting was not scoped for the initial release, a high-level overview was made in an attempt to ensure that the solution built would be extensible enough to employ one or more of the following technologies.

a) Hibernate

This software, designed for database persistence was also intended to make data extraction easier. By allowing for custom queries to be configured, a more homegrown approach to reporting could be made possible.

b) XSLT

Used in conjunction with some data extraction technology, XSLT could be utilized as a means to apply different stylesheets to a single XML document type definition (DTD) for multiple output formats. This technology was designed to make XML transformations less cumbersome and easy to maintain. XSLT allowed for the transformation of a document without the need to create a custom Java to parse, reformat, and send the contents to another file.

c) Formatting Objects Processor (FOP)

Built on top of XSL, FOP allows for one to transform an XML file into the following output formats: Portable Document Format (PDF), Printer Control Language (PCL), Postscript (PS), Scalable Vector Graphics (SVG), XML area tree representation, Print, Abstract

Windowing Toolkit (AWT), MapInfo Interchange Format (MIF), and plain text.

d) Jakarta POI

This technology was supposed to enable an application to parse and generate Microsoft Excel and Word documents.

D. Interviews

Karen Worminhaus – Business Manager/Owner

- April 2004 – Initial Contact
- May 2004 – Initial Requirments
- June 26, 2004 – Detailed Requirements I
- August 14, 2004 – Detailed Requirements II
- September 4, 2004 – Project checkpoint
- October 29, 2004 – Review of initial prototype and receipt of additional requirements
- December 18, 2004 – Prototype review and requirements clarification
- March 12, 2005 – Project checkpoint
- May 21, 2005 – Project checkpoint

Steve Troy - Director

- April 2004 – Initial Contact

- May 2004 – Initial Requirements
- August 14, 2004 – Detailed Requirements II
- November 20, 2004 – Review of initial prototype and receipt of additional requirements
- March 12, 2005 – Project checkpoint and receipt of additional requirements
- July 9, 2005 – Project checkpoint

E. Summary of Knowledge of Topic

As the student had been a professional C and report developer prior to inception of this project, his only exposure to Java, J2EE, and website development had only been primarily through school. With the object-oriented courses taken prior and during his tenure at Regis University focused primarily on foundational knowledge and general theory, this research phase of the project proved to be extremely useful. As Java and J2EE had been relatively new, these technologies were constantly evolving, and many useful tools, patterns, and frameworks had been developed to increase productivity. The knowledge obtained through this research phase helped the student make educated decisions regarding the tools and appropriate technologies required to meet Sustainable Resources' business requirements.

At a more basic design level, the developer learned the value of splitting the application into distinct tiers: presentation, business, service, and data. In doing so, it made him more responsive to the continuous changes required

by Sustainable Resources. By utilizing design patterns such as MVC, Factory/Interface, and DAO, skills obtained from these efforts helped make the student more effective in his professional career, as his employer had been moving their corporate IT infrastructure into the same direction utilized by this project.

Beginning at the data tier, the student had been exposed to Oracle at a professional level. During his exploration of the other database products available, the student obtained a better understanding of JDBC and the flexibility it allows for future changes. These data access technologies were found to be light years ahead of what the student had been exposed to as well. As he had become an expert at writing SQL, it had also become apparent that all SQL was not equal. Therefore the power and necessity of ORM technologies became readily apparent for their abilities to generate SQL appropriate to a specific database without the developer needing to worry about each vendor's individual implementation.

However, with all of the new tools available on the market, the student found that there was not any proverbial "silver-bullet" that would do everything for him. Even though some tools offered some powerful features, it was not always wise to use them. "Do not use a Cadillac when a Pinto will suffice," as Robert Sjodin once told him. The moral of that story is to architect a solution using what is necessary instead of the newest and greatest toy.

F. Contributions to Field

Although this project may never be deployed due to organizational issues with the non-profit organization for which this application was built, the skills obtained by the student during this entire process and time spent at Regis University had been reaping benefits for the student's employer. With the knowledge gained of J2EE technology and a number of related tools throughout this endeavor, the student had secured a more influential position utilizing these skills. As they had been in the process of re-architecting their systems using an OO approach and most of the developers and architects in his shop were skilled primarily in C and PL/SQL, the student had been placed in situations where he could maintain several of their current J2EE applications. Additionally, he had been tasked to evaluate tools and processes that could assist with this transformation.

At a more academic level, the student may decide to scale down this application and share it with the open source community. By removing many of the conference centric aspects, the content management portion of the site could be marketed as a lightweight, low-cost, and easy to maintain web site solution. Requiring only servlet container and database, both of which can be obtained free of charge, a business could potentially design their entire website with minor IT support while ensuring they are maintaining good navigation practices merely using a web browser.

III. METHODOLOGY

A. Life Cycle Model

At the inception of this project, the Rapid Application Development (RAD) approach to development had originally been envisioned. This choice was made for its history of building quality applications quickly and efficiently. However, this approach required much input from the business during the development phase in hopes to increase the chances that this application met Sustainable Resources needs.

In an attempt to facilitate this approach, much effort was taken during the research phase to find the proper Computer-Aided Software Engineering (CASE) tools. In addition, the student purchased his own Linux server, as that was the platform in which the application would eventually live, despite recommendations that Sustainable Resources provide one at the project's inception. This was done so members of the organization could have 24/7 access to the application in order to provide input regarding functionality and usability. After the applications infrastructure had been built and core client functionality was developed, the application was deployed to it, however, members of the organization never made use of it.

As the RAD methodology is most effective with consistent collaboration with the business, the Software Development Lifecycle (SDLC) had been effectively degraded to more of a Waterfall methodology. The student felt that Sustainable Resources should have been more engaged in the process. As they do not employ full-time employees, often times it was difficult to contact

members to answer questions. This is not meant to place the onus completely on them, as the developer was sometimes hesitant to contact the organization due to the continuous requirement changes, as they never truly knew what they wanted and the developer did not fully understand and all that occurred at the conference.

B. Development Software

As RAD generally works the best using development tools designed to increase productivity the following tools were chosen to help facilitate development.

1. *Eclipse 3.0* Integrated Development Environment (IDE)

Eclipse 3.0 was chosen as IDE due to it being a high-quality open source product, acceptance amongst the development community, extensibility provided by the many available plugin products, and its use at the student's employer. Most importantly, as this is a free product and the non-profit's IT budget was nearly non-existent, future developers could easily maintain the project.

2. *SDE* Unified Modeling Language (UML) Tool

SDE Professional Edition by Visual Paradigm was chosen as the modeling tool. At the time of purchase, it supported the UML 2.0 specification, and later an upgrade was available which allowed support for UML 3.0. As the criteria for choosing such a tool was rather stringent, this was the biggest expense incurred by the student during the project, which could be considered a problem going forward, as the non-profit

organization would have needed to pay over $800.00 to obtain a license. However, as the developer required that the product perform round-trip development in *Eclipse 3.0*, this was the highest-quality product available at the lowest cost. Rational's XDE product exceeded $1500.00 and Borland's Together Control Panel products exceeded 3,000.00 per license.

3. *Eclipse* Plugins

a) *MyEclipse*

MyEclipse was an organization that packaged many plugin components together into the *Eclipse* framework. This product was chosen primarily for its presentation tier components. As the site was developed exclusively using *Struts*, its "struts-config.xml" editor proved to be of some value. Although the student did not utilize its visual components primarily because he wanted to fully understand what was occurring under the covers, its context checking was found to be very useful.

The other feature used extensively within this product was its JSP editor. Although it was nowhere near as complex as the product provided by NitroX, *MyEclipse* was much more cost effective. Not having the WYSIWYG capabilities was not found to be a problem, however, as the student wanted to gain a greater understanding of JSP, thus, found coding the pages by hand a much better approach to learning. *MyEclipse's* abilities for context checking of standard JSP, JSTL, the *Struts* tag library, and custom tags proved to be invaluable.

b) *Clay* Database Modeling

Used throughout the entire lifecycle, this tool allowed for the database to be designed and built visually. The resulting entity relationship diagram (ERD) was then converted into the SQL data definition language (DDL) based on the chosen database. As it was not the goal of the student to learn how to administer databases, this product allowed him to not have to be overly concerned with having to become proficient in yet another language.

c) *Quantum* Database Viewer

This tool was utilized for viewing the contents of the, issuing queries, and inserting data into the database. It was also used extensively during unit testing. Its querying abilities took the form of a wizard or by issuing SQL commands directly. The developer often performed inserts into the database using this tool each time it was rebuilt load the application's initial values that it required to run.

d) *Hibernate Synchronizer*

As it was decided to utilize *Hibernate* as the application's persistence mechanism, the creation of the descriptor files required by that tool concerned the student, as he had never utilized it and feared that debugging the hand-created descriptor files would be difficult to debug. During research into *Eclipse* plugins, he had run across the *Hibernate Synchronizer*. Although this tool created files requiring some massaging, it accessed the database, read the columns and foreign key constraints of the selected tables and generated all of the

necessary mapping files necessary to have *Hibernate* functioning in a very short order. The best feature of this tool was its ability to understand the relationships between the tables and properly map them. Although some were not implemented, it went a long way to assist the developer's understanding of these relationships and made it possible to create Java objects that *Hibernate* would eventually end up populating.

4. *JUnit* Unit Testing Tool

Already an industry standard with full native support within *Eclipse*, *JUnit* was utilized during the development of the business, service and data access layers of the application. As the developer scripted calls to each business object that accessed a DAO objects, he could test each layer every time a change to the data model was performed ensuring other functionality did not break.

C. Architectural and Deployment Software

1. *Ant* Build Tool

Ant, developed by *Apache's Jakarta* project, was utilized as this project's build tool. As it is an industry standard with native support inside the *Eclipse* toolset, the student did not even try to find any competing products.

2. *Tomcat* Servlet/Application Server

As this purpose of this project was to create a Java-based web application using an open source server, only two products were seriously considered. The student originally felt that he needed to utilize *JBoss,*

however, after fully evaluating the business requirements, it became apparent that *Tomcat* would be more than sufficient to deliver the requested functionality.

As CMP EJBs had been quickly losing acceptance in the IT community, it was decided to take another approach to persistence. Even though building the application's business tier using Session Beans could have given Sustainable Resources greater flexibility in their systems, it became apparent that taking such an approach was overkill and the decision was made to build a more custom approach to that tier.

3. *MySQL* Database Engine

MySQL was an easy choice of database engines. As this was the database currently in use by Sustainable Resources' current web hosting company, this free product was also available for all major platforms free of charge, which made it possible for the student to develop the application on his Windows based machine and port it over to the provider's Linux server.

For a free product, *MySQL* was found to be extremely robust. As the data model for this project contained over 30 tables containing many foreign keys relating to other tables or back to themselves, its support for these types of constraints were a necessity. Additionally, as the data model had been designed to contain historical data, it could potentially grow quite large. Therefore, *MySQL's* support for placing indexes on

selected columns would help to ensure the database would not lose much performance as the tables grew in size.

4. ***Hibernate* ORM Tool**

After using the *Hibernate* ORM tool, the student did not have to write a single line of SQL used by the application other than the initial data definition language (DDL) and data modeling language (DML) SQL required to load the database with the minimal data required by the application.

In choosing *Hibernate* as the applications persistence mechanism instead of heavier-weight alternative, EJB, the application did not require the heavier-weight application server, *JBoss*.

D. Third Party Software

1. ***HardCore Web Content Editor* for Content Management**

Initially, Sustainable Resources wanted to retain their use of the *Macromedia Contribute* product, however, after they added the requirement of supporting multiple conferences and maintaining an archive of past ones, it became obvious that utilization of static HTML pages would no longer meet their needs. Additionally, as it was found users could easily get lost in their spaghetti of pages and disparate hard links throughout their site, it became obvious that a more sophisticated approach was required.

As these functional requirements were defined well into the development phase of the presentation tier, the student hoped to find a

Commercial off-the-shelf (COTS) solution to meet the organization's needs. However, as the developer did not have any budget offered by the organization, he needed to either find an open source solution or build one himself. The only COTS solution found that meet this budgetary requirement was *OpenCMS,* however; after many failed attempts to get this product to function; it was decided to build a custom content management solution.

The solution architected was built on top of the *Struts* framework already leveraged within the application. The developer designed the solution by creating a single JSP page that obtained its template definition and contents from the database. The final piece required an HTML editor that would function within a browser window. After looking into several products, the *HardCore Web Content Editor* was chosen primarily due to its low cost (approximately $40.00 which was never paid by Sustainable Resources) and ease of use. It allowed for the generation of professional looking web content without requiring the user to learn HTML.

2. *LinkPoint* Credit Card Processing Software

After reaching many dead-ends during research into credit card processing in J2EE applications, the student contacted the company that held Sustainable Resources' merchant account. Their systems were setup to accept transactions via LinkPoint, a company that provides E-commerce Internet services. (Note: the XML message required by the

receiving entity had been formed in preparation of the eventuality that the organization would finally purchase the API)

E. Specific Procedures

In an attempt to stay true to the RAD approach to design and development, the student attempted to keep the organization abreast of all recent developments by setting up meetings, emailing diagrams, and even providing a server containing the latest version of code so they could check the progress and suggest any changes. The student always initiated these contacts however. Although Worminghaus was very helpful throughout the early phases, the director of the organization was never fully engaged with the process nor appeared to fully understand the entire scope of the project and the SDLC as a whole. As stated earlier, the RAD methodology deteriorated somewhat into a version of the Waterfall method in which documentation was delivered at the end of each development stage.

F. Review of Deliverables

Use Case and Activity UML diagrams were produced during the Functional Design phase, and the Technical Design phase produced Class and Sequence diagrams, which were reviewed by Worminghaus. After the development phase commenced, the deliverables consisted only of prototypes. Although a creation of a system test plan was originally scoped, the organization never produced any resources as originally agreed.

G. Resource Requirements

Other than a business owner dictating the organizational needs, Sustainable Resources did not invest anything into this project although they were asked well in advance to provide a test server and purchase the application's third-party software. The student provided all other of the required resources for this project. Fortunately, however, most of the software utilized for this project was free with the exception of the package utilized for design, which was expensed through the student's employer as an educational expense. Regarding interfacing with the organizations ISP, the student tried to contacting them regarding data migration, but never received any calls back, however, as the organization lost complete interest in the project, this issue became moot.

H. Outcomes

Considering that Sustainable Resources never took ownership of this project, a highly functional application had been built. All may not be lost regarding its deployment. Sustainable Resources at some time may potentially revive its annual conference, however, they still must provide testing resources and take a more active role in the project. The only other way the student will release this application is if it is purchased, and any additional development required including the implementation of security would no longer be free-of-charge.

I. Summary

Although RAD was not fully implemented during this project, achieving an understanding of the tools and processes required of this methodology was a valuable lesson. As the student's employer is looking for ways to move away from the Waterfall into a more iterative SDLC, extensive knowledge was gained through the review of the many CASE tools on the market and use of others.

IV. PROJECT HISTORY

A. How the Project Began

After exploring numerous project ideas over the last two years, each had been systematically eliminated for various reasons. The primary goal of this search was to find a non-profit organization or educational entity without much funding that required an application to increase their productivity. It was also very important that the organization shared many of the student's core values.

A day before consulting with Regis' Professional Projects administrator, Tricia Litz came into contact with Steve Troy of Sustainable Resources. He was requesting some help with maintenance of their previous site. After the student described his intentions, Litz felt that this relationship would have been a nice fit.

B. Project Management

1. Analysis and Design

Analysis of Sustainable Resources' existing web solution began in April 2004 after the student's initial contact with the director, Steve Troy, and their business manager, Karen Worminghaus. At that time, the project was primarily transferred to Worminghaus, as she was responsible for managing the details of the conference such as: presentations, sponsorship, pre/post conference workshops, volunteers, attendees, and all of the financial transactions attached to it. Originally, the proposed

application was required to only handle these transactional aspects for a single conference.

As expected, minor changes to the original design were made throughout the initial design phase. Many of these issues were due to the student's misunderstanding of how the conference was structured, and most of the resulting changes were relatively easy to implement. By the end of 2004, an initial prototype that implemented most of the required functionality including: user registration and maintenance, the ability for individual users to request different participation types, and the ability for the application to bill the user. Some administrative functions had also been implemented such as: the ability to change some display values stored in the database and change the prices of the different participation types and levels. As for the static content, the student began migrating their static pages over to the application to help round out the initial demonstration version.

After demonstrating the application to the organization, the student communicated that he intended to fully implement the same functionality that the registered user had to the administrative screens. However, at this time, the organization dropped several new major requirements upon the student. Initially, the student communicated that these changes would have to take place in a subsequent release, however, after analyzing each of the requests, they had such profound impacts to the overall application that it would be more advantageous to implement them right away. This

development effectively pushed the expected release out by approximately four to six months.

Management of Multiple Conferences

As the initial requirements stated the application be responsible managing a single conference, the late request for management of multiple conferences at multiple sites impacted nearly the entire data model. As the user interface was still in its infancy, the presentation tier required little refactoring.

Content Management and Conference Archive

Although it had been decided earlier that Sustainable Resources would continue managing their site's static content manually using *Macromedia Contribute,* the number one priority of the application had shifted from conference management to content management. Driven primarily by the new requirement of management of multiple conferences and the necessity of maintaining an archive of selected pages, it became necessary to determine whether or not to find an open source content management COTS package or build a custom solution. After a couple weeks of analysis, it was decided to use the existing *Struts* layer in conjunction with a third party Javascript enabled HTML editor that could function within a browser window.

a) High Level Functional Requirements

The application was effectively split-up into three distinct areas: general user, registered user, and administrative user. The functionality of the top level was inherited down the chain.

Anonymous User

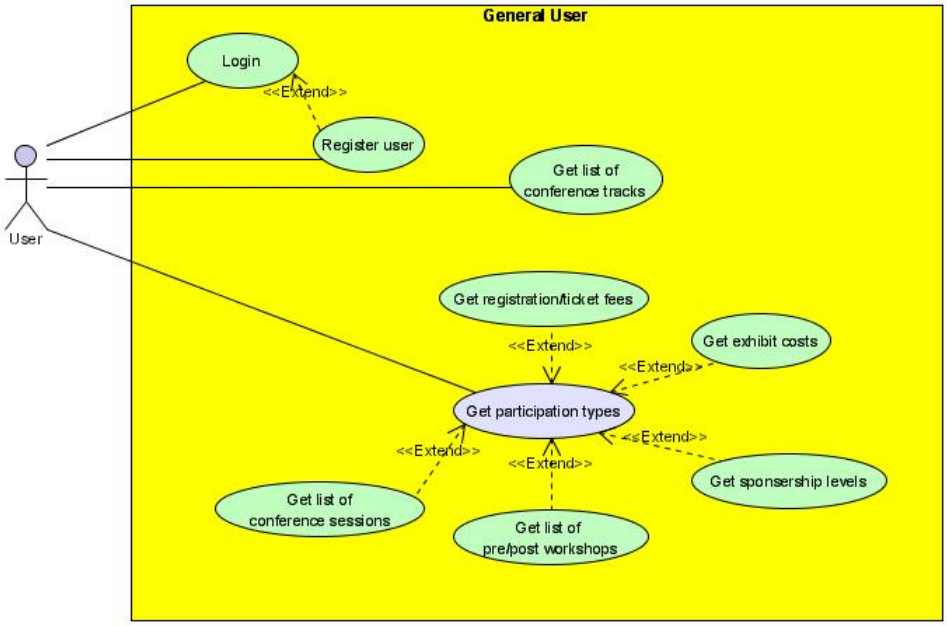

Figure 1 – "General User" Use Case Diagram

Upon entry to the site, all users were considered "Anonymous" until they either logged in or registered. These users only could view the "static" content for either conferences defined as active or ones that have been archived. The user could select the conference to view via a link rendered at the bottom of the home page. Each page had four separate menus and a link to a site map all designed for ease of navigation throughout the site. As the templates of the pages were stored in a hierarchical manner, the program responsible for rendering the site map needed only to recursively loop through each branch to display the structure of the entire "static" portion of the site.

Top Menu

This menu contained within the banner atop the page was static in nature and was designed to contain links to some of the most visited pages.

Left Navigation Menu

Like the top menu, this left navigation menu contained several static links to pages such as “Login”, “Register”, however, each individual page could be configured to have this menu change at runtime. As the pages were organized in a hierarchical manner, another template attribute needed to be added to the currently displayed page’s template. Displaying all of the first-level children of this attribute generated the rest of the links.

Sub-Navigation Menu

This optional menu was designed to sit directly above the page’s contents. Using the same pattern employed in the left navigation menu, this menu was designed to display links to each of the currently displayed page’s siblings.

Location Navigation Links

Located above the sub-navigation menu, the location navigation links displayed the depth at which the user currently was within in the page hierarchy. This would allow the user to quickly traverse up the tree. Although these links were generated dynamically, the pattern employed here did not require any additional attributes to the page template object, as traversing the up created them the tree until the top level was found created these links.

Page Content

Each page contained multiple blocks of content in the form of an HTML fragment or the inclusion of another JSP module designed to perform some dynamic action such as displaying some data contained within the database. The main use for this facility was to return conference information such as the different tracks and participation types and costs associated with each. (Note: when an anonymous user tried to request a participation type, they would be prompted to either login into or register with the system)

Registered User

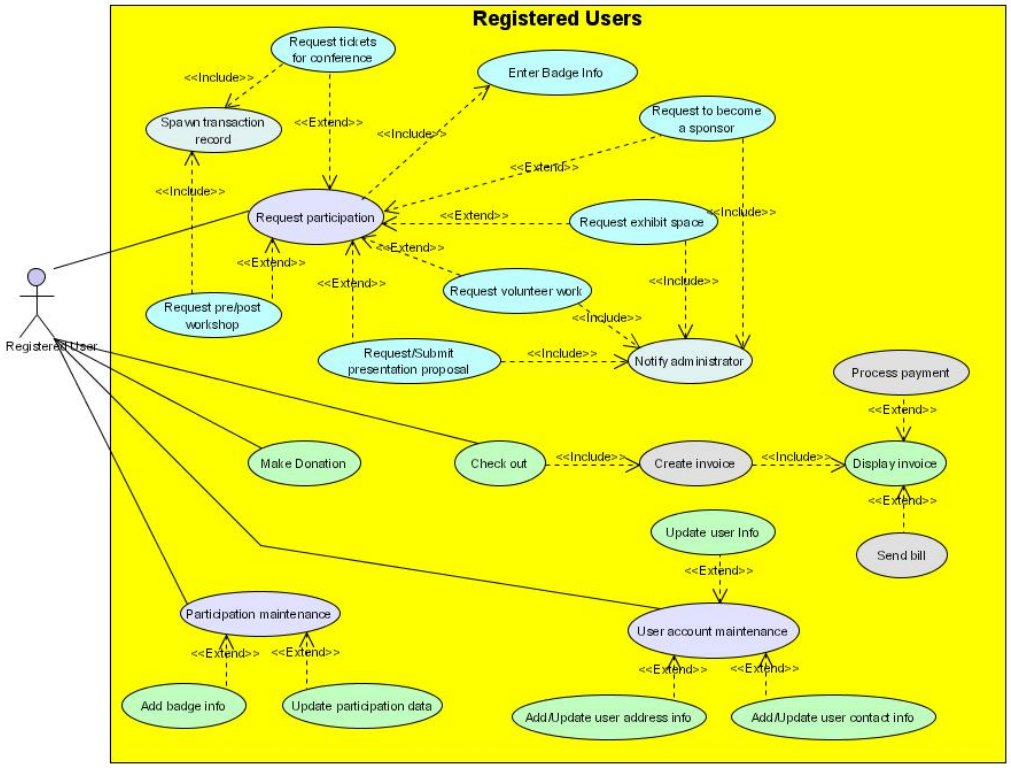

Figure 2 – "Registered User" Use Case Diagram

By registering or logging into the system, the user automatically would be eligible to sign up for and maintain their conference activities, make donations to the organization, manage their account, and make payments. Page-level security had been added to all of these pages. In the case a person's session either expired or a link to a page accessible only to registered users was directly accessed, the user would be prompted to either log into or register with the system.

Administrative User

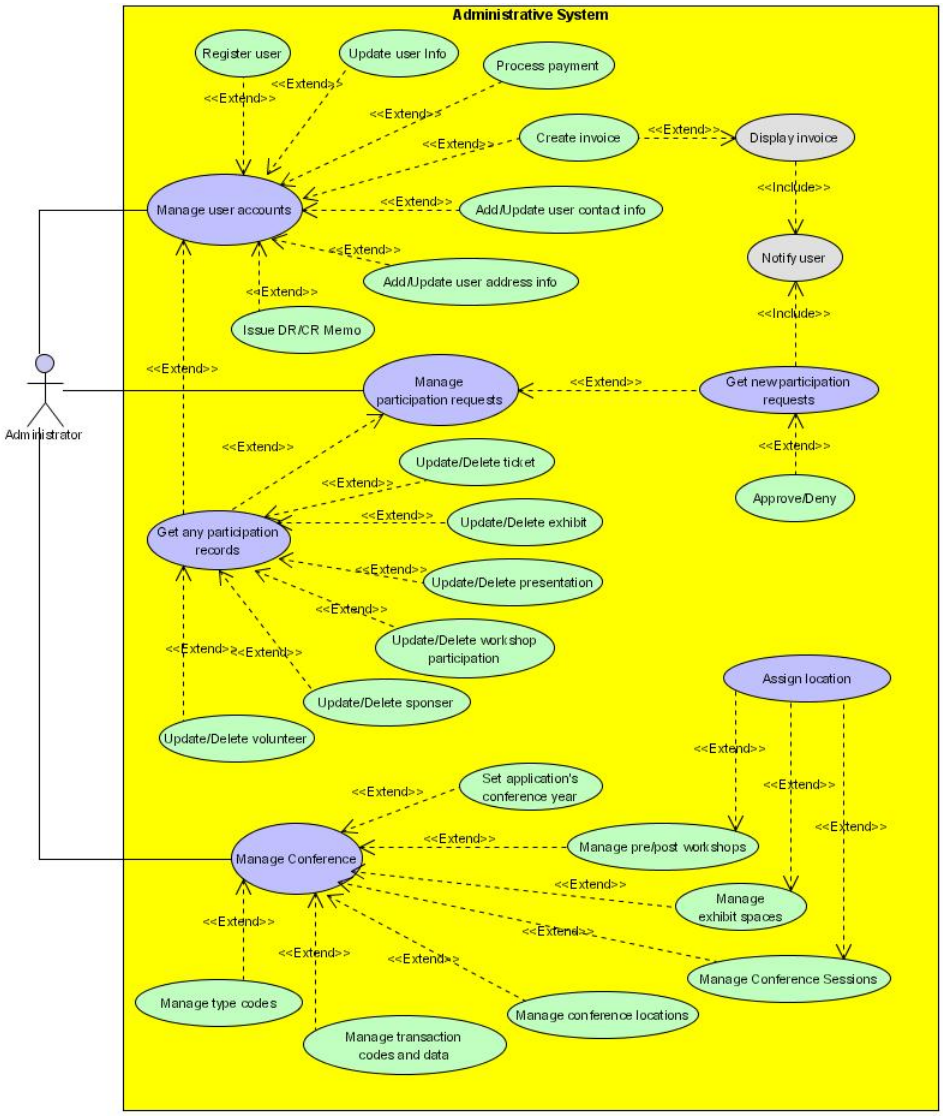

Figure 3 – "Administrative User" Use Case Diagram

An administrative account had control over nearly every aspect of the site, and was the most difficult portion of the application to develop. Like the pages associated with a registered user, the administration pages had the same type of page level security employed. An administrator's capabilities had to be broken into five separate categories: Manage Conference, User Maintenance, Participation Maintenance, Code Maintenance, and Content Management.

Manage Conferences

Within the "Manage Conference" link, the user had been given four options. The "Manage Conference Sites" link allowed the user to either enter a new conference site or change its type code or description. "Conference Management" allowed the user to either create or edit attributes associated with a conference record: conference site, name, begin date, and end date. "Set Default Conference" defined the conference record that was loaded when a user first entered the site by use of a drop-down list box that displayed only active deployed conferences. And finally, the "Set Current Conference" option utilized a drop-down list containing all active conferences and set the selected one into the session so all subsequent operations would be performed to that particular conference.

User Maintenance

This option allowed the administrator to perform all user related functions such as registration and management of a particular user

account by selecting from a list or searching by last name. The administrator also had the ability to change a user's status and type, which a normal user could not do. Additionally, several links were provided on the "Account Information" page that linked directly to the user's transactional records: Find Participations for, Create Participation, and Bill User.

Participation Maintenance

This section contained four sub-sections: "Find Participations", "Manage Locations", "Manage Workshops", and "Manage Conference Sessions". The purpose of "Find Participations" was to search for participations by type and stage so the administrator had the ability to change the stage or attributes of a participation. "Manage Locations" was designed to creates and assign location records for exhibits, conference sessions, and workshops. "Manage Workshops" maintained Pre/Post Conference Workshops. And finally, "Manage Conference Sessions" was responsible for grouping together participations and assigning a location.

Code Maintenance

The "Code Maintenance" interface traversed through the Type classes designed primarily for use as meaningful labels. As these objects were stored in a hierarchical manner, the interface allowed the user to drill into the hierarchy to find the code requiring maintenance. In addition to managing types, as a "Transaction Type" was an

extension to a Type, the administrator could also manage participation prices through this interface as well.

Content Management

When logged into the system as an administrator and accessing the “static” portion of the site, a link would appear at the bottom labeled “Edit Page”. When clicked, the application would open a page containing four page attributes and an HTML editor in which the page content could be edited. After the “Preview” button was pushed, the application would render the changes to a preview page where they could either be accepted or rejected prior to be committed to the database.

The four other attributes mentioned earlier set other aspects of the generated HTML. The first “Page Title” was designed to set the <title> tag within the <head> portion of the document. “Page Heading” was displayed at the top of the content portion of the page. Finally, the two input fields, “Metadata Description” and “Metadata Keywords” were not meant to be displayed anywhere but designed to hold would-be data gathered by web crawlers.

Additionally, when accessing the site map, the ability to add, edit, and remove pages within the hierarchy had been implemented. While adding or editing a “Page Template”, the user could define the following attributes: “Page Title”, “Page Heading”, “Sequence”,

“Number to Archive”, “Archive Page”, “Link Page to Conference”, “Default Page”, “Root Subnav Page”, and “Root Left Menu Page”.

“Sequence” was designed to define the position in which it would be displayed relative to its siblings. “Number to Archive” was designed to hold the number of content pages that would be persisted. “Archive Page” determined whether or not this template page was added into the archive portion of the site, “Link Page to Conference” (available only at the time the page is created) check box defined whether or not the contents of a page would be tied to a particular conference or globally.

The “Default Page” attribute only appeared when editing a page. When not null, a template page with this attribute set to “true” would become a logical entity, meaning its contents would be contained within the chosen template. All of the selectable templates consisted of only the template’s children records. The final two attributes: “Root Subnav Page”, and “Root Left Menu Page” had been designed for the dynamic generation of the left and sub navigation menus.

b) Activity Diagrams

The Activity Diagrams contained in Appendix A depict some of the functionality designed during the initial design phase and reflect only the application’s original requirements. As the scope of this project grew so large, the student was unable to continue maintaining these diagrams due to time constraints.

2. Technical Design Overview

a) Data Persistence Tier

The data model proved to be one of the most challenging aspects during design. By taking special care to ensure all classes and tables were properly designed, the impacts of the changes required became less troublesome than the student originally had feared. Additionally, by using *Hibernate* as the persistence mechanism, data model changes impacting minor functions became very efficient. The UML Class and Entity Relationship (ERD) diagrams contained within Appendix B helped to illustrate the size and complexity of the application as a whole. In using 28 different relational tables and 30 associated Java classes plus two interfaces, the application was able to perform each of the functions outlined in the previous section.

Hibernate Implementation

Hibernate's powerful capabilities supported several types of uni-directional and bi-directional relationships including: one-to-one, one-to-many, many-to-one, many-to-many (although not recommended). The one-to-many and many-to-one were extensively utilized throughout this application.

Two main ways of depicting inheritance were utilized by this application as well. The Payment classes (Cash, Check, and Credit) utilized the "subclass" methodology, which meant a single table utilizing a particular "discriminator value" determined which class type was inserted or would be instantiated during a select operation. The

main problem with this approach was that nearly all of the columns needed to be defined as optional and the program logic would need to programmatically enforce required fields.

The most complex inheritance model, however, was implemented with the Participation classes. Using the “joined-subclass” approach, a table needed to be generated for each class, including the abstract “Participation” class (due to periodic instantiation exceptions the “Participation” class, the “abstract” modifier needed to be removed from the class definition). Each of the subclass tables required a primary key, which was actually a foreign key back to the primary key column of the Participation table.

b) Sequence Diagrams

The application’s Sequence Diagrams were created immediately after the data tier was designed and have been placed into Appendix C. However, as with the Activity Diagrams, these only depict the objects and methods of the functionality designed just after the initial requirements and were not updated since due to time constraints.

c) Service Tier

The service tier had been developed by means of the Factory/Interface pattern. This approach was implemented by the business service (UserMgr in the illustration below) calling the ServiceFactory to receive the configured instantiated class that implemented the required interface. In order for the ServiceFactory to

dynamically instantiate the required class, it was required to retrieve the data contained in the “Service” table. To reduce the amount of coupling between tiers, methods within these services only focused on performing a single function well.

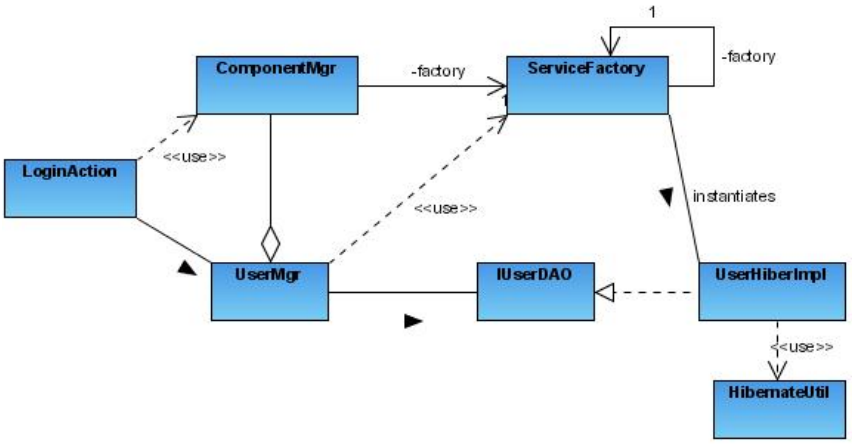

Figure 4 - Business and Service Tier Patterns

d) Business Tier

Nearly all of the business logic retained in this application resided inside the classes within the “org.sr.conf.business” package and were all suffixed with “Mgr”. All of these classes defined within the “Manager” database table were instantiated each time a person entered the application and were stored inside of the ComponentMgr object and saved within the user’s Session object. Each time either a *Struts* Action or Taglib class needed to access the business tier, the ComponentMgr was retrieved from the HTTP session and the business object was returned to the caller via its key.

e) Presentation Tier

At the inception of this project, the student felt that the service tier would have been the slowest and most difficult portion of the application to develop. However, once he began creating all of the HTML forms required for data entry, it soon became apparent that there were many more aspects to manage. Additionally, as administrative users were able to perform the same functions as a registered user but as a proxy, good design would dictate that these components be reused.

Struts Configuration

The backbone of a *Struts* application is the “struts-config.xml” file. It contained most of the URL mappings, form bean definitions, and required actions to be performed on the server.

Form Beans and Validation

Struts gave the developer several means to create form bean objects. *Struts* automatically populates these objects so the HTTP message need not be manually parsed thus increasing productivity. This application utilized DynaActionForm and its derivative called DynaValidatorForm. In using these constructs, the student needed only to define these beans within the “struts-config.xml” file without creating a new class, which was performed automatically by the container.

As user input validation had always been a tedious task, the student made full use of *Struts* built-in validator capabilities to help

expedite this exercise. This robust library contained many of the most common input types, but also allowed for custom validations to be defined. Some of these validations developed accessed the applications business tier for reading other values, or created date objects wherever multiple date objects had been entered.

Action Classes

Once data validation had been performed, the contents of the form bean would be extracted and sent along to the business tier for processing. As this application performed similar functions regarding user maintenance at the registered user as well as at the administrative user areas, some of these action classes had to know who was calling it and why. Once this data was defined, it was then possible to use modify that Action to ensure an “Administrator” would not get returned back to a “User” screen.

Custom Tag Libraries

Struts contains a robust set prepackaged tag libraries, however, like the validation routines, it was not always possible to utilize these tags or the ones located in the JSTL. For this application, the developer decided that the creation of an Action class each time some data was required from the database would have been overkill. The student also took the approach where each page needed to be responsible for its own display. Therefore, he decided to implement specialized tags that connected to the business layer for the retrieval of data.

Tiles

This aspect of *Struts* was, without a doubt, the most important feature utilized that fostered reuse. The whole idea around tiles is that a JSP template be created containing keys to a tiles configuration file that linked a page with a series of other JSP pages (configured within the “struts-config.xml” file). By taking this approach, the given template would define all of the static aspects of a page such as the: banner, side menus, top menus, footers, etc., as each page definition could inherit the properties of any other. Only the JSP file containing the logic to render the dynamic information needed to be created. As this application was developed with three distinct areas in mind, the student created three different tiles definition files and template JSP files.

C. Significant Events and Milestones

1. CASE Tool Research

Occurring during the requirements gathering phase, the student spent many hours reviewing products as the goals of the organization began to be fleshed out. Even though it was the student’s goal to produce a J2EE application and wanted to focus upon the tools being leveraged at his employer, this time was decidedly well spent, as some tools initially decided upon were dropped for others. The selection of a UML tool supporting round-trip development was key, as it was found to increase productivity. Not only did it create functional code, but also the act of visually designing the data objects allowed for this tier to be effectively

refactored, as it created empty classes also saved the student many hours.

2. **Functional Design**

Functional design basically occurred in two distinct phases. Initially, after working with Worminghaus on the transactional aspects of the application, design of the data model commenced. Even though minor changes became necessary while the project was in the technical design phase primarily due to the student's lack of understanding of the business, being able to iterate to prior phases is the main premise of the RAD methodology.

3. **Technical Design**

Technical design proved to be the most challenging aspect of the project. Initially, the creation of a data model that could meet the organization's needs for the project and beyond..

4. **Completion of Development**

Occurring in August 2005, completion of the development phase had been long overdue, as an initial release, including only some base functionality had originally planned to be made just after the fall 2004 conference. However, as the organization was in no hurry to receive the application, it was agreed by both parties that it was more important to fully implement nearly all of the requirements prior to release.

5. **Professional Project Presentation**

Although the student was given the option to present this application to Regis remotely, it had been decided that it would be more meaningful to

demonstrate some of the main functions of the application to the board. The student was especially proud of the creation of the content management system.

D. Changes to Project Plan

The two main changes to the project plan were: the additional requirement of handling multiple conferences and creation of the content management piece. However, as the student was planning on having members of the organization more involved, he was hoping that he would have had much more feedback on the prototypes deployed to the test server. Without that valuable input, it was difficult to assess whether or not the application, especially the user interface would meet the organizational needs.

E. Evaluation of Project Goals

As the original plan was primarily to create a transactional site to manage a conference and any additional requirements would be supported with in future releases, the goals set forth by the student had been exceeded. Outside of exceeding the functional requirements spelled out by Sustainable Resources, the student learned much more than he had planned.

As the student had never created a Java/J2EE application prior to his work on this project, through each step: from the research put forth into development tools to the work required for the content management piece, the knowledge gained through this effort became far more robust than had been envisioned. Technically speaking, however, the two biggest goals not

achieved while developing this project was the implementation of SSL on all of the site's forms and credit card processing even though the message to their credit card processing company had been formulated.

F. What Went Right/Wrong

As with every development project, many coding issues came up along the way, many of which were due to the student extending known and learning new technologies. The largest success of the project would have to have been the flexibility and functionality of the persistence layer. Much thought and effort was put into creating a data model that could be easily extended and would will with an ORM tool such as Hibernate. Without this success, the highly functional participation and content management portions of the application would not have been so successful.

Other than the well-documented problems encountered with the student's dealings with the organization and the extended timeline necessary to meet their needs, each phase took longer than originally anticipated. As many at the student's place of employment view him as a "glass half-full" sort of person, his generally positive outlook can become a negative at times, as it affects his ability to properly estimate timelines. One other thing, which could be considered a negative, was in regards to the design of the business layer, as it had some minor coupling with the *Hibernate* persistence layer.

G. Project Variables and Impacts

As Java and J2EE technologies had been constantly evolving, difficult choices were required in choosing the proper toolset of the numerous

available played a key in the project's overall timeline. As it had been customary to build applications using a plain text editor in the past, the new complex and extendible IDEs available allowed the developer to create this large application much more quickly than had been possible in the past. By having the ability to have integrated modeling tools, real-time syntax checking, code generation, build tools, server utility programs, etc. all within a single application, many command-line and manual tasks, which had been a tedious aspect to programming in the past had largely been eliminated.

The utilization of other utilitarian types of programs such as *Struts* to help facilitate design patterns like MVC were also found to be invaluable. By removing the necessity for a programmer to write code to perform tedious tasks such as parsing through an HTTP request and input validation, the productivity gains were so vast that they were incalculable. Additionally, by leveraging many industry standards, the student felt that the application would be more maintainable, as future potential programmers would understand the application's inner-workings much more quickly.

H. Analysis Results

All said, despite the student's problems with Sustainable Resources, he was extremely proud of each section developed. Although never fully system or load tested, it functioned well, and the input validation routines employed were very effective and provided meaningful.

I. Summary of Results

The 15 months of development were wrought with just as many trials as successes. By taking the approach of developing an application to solve real-world problems, the student was able to experience the many issues inherent with the SDLC. From the analysis through the implementation phases many decisions had to be made by the student that may not have been experienced by working in a practicum. However, the experience missed the most by taking this approach was working with a team and drawing from others experiences, which could have smoothed many of the tribulations encountered.

Although each phase took the student longer to complete than anticipated, the student thought that the presentation tier would be the easiest. This misconception was the second largest reason why the student's estimates were so far off. By not being able to script JUnit tests and having to write some relatively complex input validation routines, it made the back-end design appear to be the easy phase. However, had the analysis focused primarily upon the UI, more changes to the services would have become necessary, thus prolonging the UI development even further.

V. LESSONS LEARNED

A. Lessons from Project Experience

The biggest lesson learned through this project is one of human nature. Even though the student had been taken advantage of enough times in the past, he had never invested so much time into something just to be ignored after a task had been completed. To help avoid such issues in the future, as the student would like to again offer his services to worthy organizations, the first thing he would do to ensure that the organization benefiting from such labor had some up-front investment something into the project as well. Additionally, agreements in writing should be made outlining the expected deliverables and the scheduling of regular meetings to make they do not lose interested. As the student invested vast amounts of time, energy, and money into this project it was a shame that the delivery of much more functionality than required for satisfaction of the student's academic achievement may never be utilized by the organization for which it was built.

1. Project Management

Although not originally sought out, the student had gained a higher respect for the work performed by project managers. This unenviable job, which entails the gathering and orchestration of resources, takes a special type of person. Being more technically minded, the student did revel in the tasks required to ensure the people within the organization were kept abreast of the project plan.

2. Analysis and Design

a) Requirements Gathering

The biggest lesson learned while gathering requirements was to let the business know that they need not concern themselves with how the functionality would be implemented, as they often appeared to be concerned too much about the how rather than the what. In doing so, they were able to focus upon what they wanted the application to accomplish. This mentality was due in part to the organization never working with an IT professional.

b) UML

As the student had really only created UML diagrams in the context of classes, he never developed an entire set from beginning to end. This exercise helped the student fully understand the power of these diagrams. The Use Case and Activity diagrams helped turn many of the abstract ideas being passed around into something more concrete. The static class diagrams helped the developer understand the class relationships and allowed for a flexible data model to be built and assisted greatly in the development of the database. Finally, the high-level Sequence diagrams helped the developer to define the interfaces to the business and service tiers.

3. Development

a) CASE Tools

Technically speaking, the student learned the value of CASE tools. As it was his goal to learn each aspect of the Java language the hard way by

not utilizing an IDE during his coursework, it became apparent that he would not want to take this approach for this project early on, as coding on a plain text editor can be rather slow. IDEs, however, perform many complex functions including real-time syntax checking, which were found to be invaluable.

b) Persistence

The power of ORM tools was found to be a major key to the success of this project. As both major and minor data model changes were found to be necessary during the development phase, *Hibernate* was key to enabling these changes by reducing impact across the application.

B. Aspects to Change

The only aspect of the application would need to be changed at this time is how the business layer interacts with the service/data access layer. As a *Hibernate* transaction needs to be issued prior and committed after issuing any CRUD operations, the decision to place these calls into the business layer effectively coupled these two layers. Although the two aforementioned *Hibernate* method calls did not belong in the DAO layer, the developer should have added an additional layer between the two.

C. Expectations Meet?

As stated previously, the application was built as requested, however, it may be never known whether or not it would have meet Sustainable Resources expectations, as they did not express any desire to view the finished product for reasons unknown to the student. Even though there

remain two technical obstacles to hurdle (SSL encryption and credit card processing) the student may decide to learn those aspects on his own, as they are increasingly becoming more important in the industry.

D. Next Evolution

As Sustainable Resources works with many international groups, multi-lingual support would be a good idea. One of the benefits of using the Struts framework is that minimal work would be required. It can store all of the static text information displayed on a web page and have them placed into configuration files that can be resolved at runtime.

E. Conclusions/Recommendations

Although the 15 months of design and development that went into this project will probably never see a production web server, working with real functional requirements helped push the student's design skills to their limits. As the original goal of the project was to create a transactional system to manage a conference, the constantly changing requirements helped drive home the reasons why the multi-tiered approach to development was so essential when architecting a system.

The addition of the content management portion of the site forced the student to think outside the box and develop data structures and algorithms, which will assist him far into the future. By forcing him to develop such a flexible data structure, his growth as an architect and ability to create programs that can evolve had increased.

F. Summary

Item	Project Stage	Description of Problem	Solution/Suggested Changes
1.	Planning	Finding UML and Java IDE software	The student originally decided to utilize the distribution of Rational XDE, however, after finding out that it did not support round-trip development between the UML diagrams and code, the decision to go with Eclipse 3.0 and to purchase *SDE by Visual Paradigm.*
2.	Planning	Deciding on an application server	It was originally planned to utilize JBoss, as EJBs were scoped, after deciding that JBoss was overly robust for this project and some ISPs may not support such an application, it was decided to use Tomcat.
3.	Design	Data persistence technology	Originally planned on using Enterprise Java Beans (EJB), but have determined that this site did not require such a robust technology. The solution was to implement *Hibernate* as the persistence mechanism.
4.	Design	Scope creep	Although the organization was told that many of their requested enhancements would be made after the initial release, it was decided to implement everything, as they were not in any hurry to receive the application.
5.	Design	Addition of requirement to allow application to manage multiple conferences	Addition of a conference attribute to all conference related tables leading to a much more flexible data model.
6.	Design	Addition of content management requirement	Implementation of custom-built CMS with pages that could be linked to a conference. The main table held a template of the entire site, which could render a site map that allowed for an administrator to add and remove tables.
7.	Design	Addition of archive for retrieval of content of former conferences	Addition of attribute to page template table, which denoted whether or not it would be available for archive purposed. The JSPs required to dynamically render the sites regular content had to be extended.
8.	Design	Enforcement of good navigation rules	Creation of dynamically generated menus based off of the page template hierarchy.
9.	Design	Misunderstanding of all conference types	As the data model was written in relatively flexibly, addition retrofitting the model was not as difficult as originally feared.
10	Development	Time estimates	The application took much longer to

			develop as the student originally felt that the UI portion of the site would be much easier. However, after developing the forms, validation was much more complex than originally envisioned. Utilization of the *Struts* validator classes helped expedite this process more quickly and elegantly.
11	Development	Prototypes	The student purchased a Linux machine to be used as a test server for which the organization could leverage to give the student feedback, however, they never accessed the server.
12	System Test	Never allocated any resources.	To date, this application has not been viewed by the organization even though it was agreed that they would be providing these resources. Due to this development, it is unclear whether this application will ever be deployed to a production server.

GLOSSARY

Ant – An open source, XML based Java source code build tool, which not only compiles, but also can copy, package, and move nearly any type of file within a local file system or via FTP. More comprehensive information on this product can be found at http://ant.apache.org/.

Application Programmers Interface (API) – The means that an application/program accesses the operating system or other services at the source code level.

browser – A program that accesses and displays files and other data available on the Internet and other networks. (http://dictionary.reference.com/search?q=browser)

build – The process of compiling, packaging, and moving source code along with other dependencies such as configuration files (also see Ant).

byte-code – The compiled version of Java source files that allow for execution on top of a Java Virtual Machine (JVM).

client – A computer program that can invokes other programs.

Enterprise Java Beans (EJB) – Java objects that run within a container/application server such as JBoss that provides transactional, security, threading, and data persistence services. (http://computing-dictionary.thefreedictionary.com/Enterprise%20JavaBeans)

Expresso – This Java based application development framework designed to shorten time-to-delivery of Web-based and transactional applications. The APIs are built based on open standards, which were designed to solve complex technical challenges and create loosely coupled applications. (http://www.jcorporate.com/html/products/expresso.html)

File Transport Protocol (FTP) – The network protocol that allows for the copying of files between computers accessible through a network connection.

Formatting Objects Processor (FOP) – A open source Java application that uses XSL to reformat XML data into other renditions, the primary being PDF. Other output formats supported are; PCL, PS, SVG, XML (area tree representation), Print, AWT, MIF and TXT. (http://xml.apache.org/fop/)

Hyper Text Transer Protocol (HTTP) – The protocol used to transmit files, web pages, and web components over the Internet or other network connections.

HTTP Server (aka. Apache) – This open source web server developed by Apache was designed to deliver of static web content via the HTTP protocol. According to http://httpd.apache.org/, it has been the most popular server since 1996, on which, almost 2/3 of Internet sites are using Apache.

Hibernate – This open source Java API is the most popular object/relational persistence and query service for Java. (http://www.hibernate.org/)

Hyper Text Markup Language (HTML) – A text-based language used by browsers to render graphical content generally delivered by the Internet.

interface – The point at which communications are initiated with a computer program and an entity such as another computer program, human, or peripheral.

JBoss – An application server built using Sun Microsystems' EJB specification designed to manage an application's business services, data persistence, and queuing needs. (http://jboss.org/index.html)

Jakarta Project – Maintains open source Java solutions for public distribution free of charge. (http://jakarta.apache.org/)

Java 2 Enterpriese Edition (J2EE) – A Java specification developed by Sun Microsystems to assist in the development of multi-tier, service-oriented enterprise application.

Java Data Objects (JDO) – A Java API developed by Sun Microsystems for database persistence. (http://java.sun.com/products/jdo/index.jsp)

Java Database Connectivity (JDBC) – A standard API interface developed by Sun Microsystems that allows a Java program to access any relational database that contains the proper driver. (http://java.sun.com/products/jdbc/)

Java Server Pages (JSP) – A language that intermixes HTML and Java technologies and allows for dynamic web page creation. JSP pages must be served up using a standard servlet container such as Tomcat.

Java Virtual Machine (JVM) – A program layer that resides above the operating system level that runs Java byte-code and is the key feature of the Java programming language which makes it platform independent.

Javadoc – Linked HTML documents that outline classes and their public interfaces and are generated directly from the code.

Model-View-Controller (MVC) – A pattern for designing an interactive application. The model denotes the internal workings or application's services. The view and controller refers to how the user sees the state of the model and changes its state by providing input respectively.

MySQL – A popular open source database. (http://www.mysql.com/)

object-orientated – A programming paradigm based on the concept of encapsulating a data structure with associated actions called methods.

open source – Software that is distributed free of distribution restrictions. This does not necessarily mean that the software is free of charge, as there are a number of different open source licenses available.

prototype – A partially working model of the user-interface that the customer can access for user acceptance testing during the development phase of developing an application.

Rapid Application Development (RAD) – An iterative software development methodology, which attempts to quickly deliver prototypes during the development stage to the customer for testing. The goal of this methodology is to deliver an application that fully meets the customer's needs quickly with minimal defects.

relational database – A database that contains multiple tables consisting of rows (records) and columns (fields), which can be related to each other.

Secure Sockets Layer (SSL) – A communications protocol built on top of HTTP and TCP/IP allowing for secure transmissions that utilize the public key paradigm and RSA 128-bit encryption.

servlet – A Java program that runs inside of a web server and responds to HTTP requests from clients.

Structured Query Language (SQL) – A standard database language utilized for creating, updating, or querying a relational database.

Struts – A flexible control framework based on Java standards such as servlets, XML, and JavaBeans to help enforce the Model-View-Controller design paradigm. (http://jakarta.apache.org/struts/)

system test – An environment that mirrors the production environment as closely as possible utilized for testing an application prior to release.

Tomcat – A servlet container that is used to send and retrieve HTTP/HTTPS transmissions over a network connection. It is generally used for the

generation of dynamic web pages using the Java Server Pages (JSP) technology.

Uniform Resource Locator (URL) – An address of a document or resource located on the Internet or an Intranet.

unit test – The act of testing individual pieces of software by a developer during the development phase of an application.

web server – A process running on a server to send and receive HTTP transmissions over the Internet or an Intranet when utilizing a URL.

XDoclet – An open source code generation engine. (http://xdoclet.sourceforge.net/xdoclet/index.html)

APPENDIX A – ACTIVITY DIAGRAMS

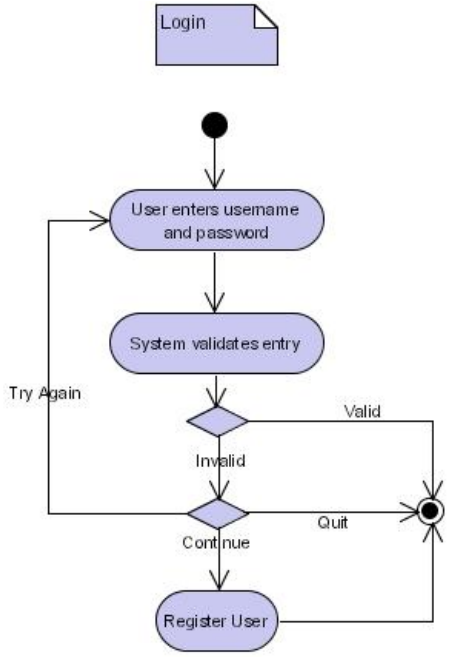

Figure 5 – Activity Diagram for "Log In" Use Case

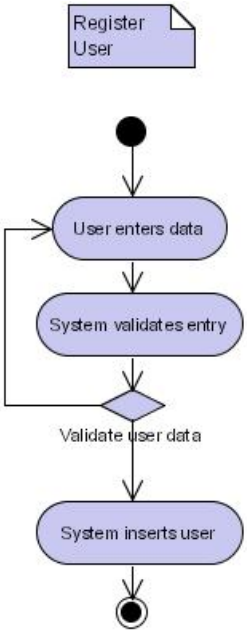

Figure 6 – Activity Diagram for "Register User" Use Case

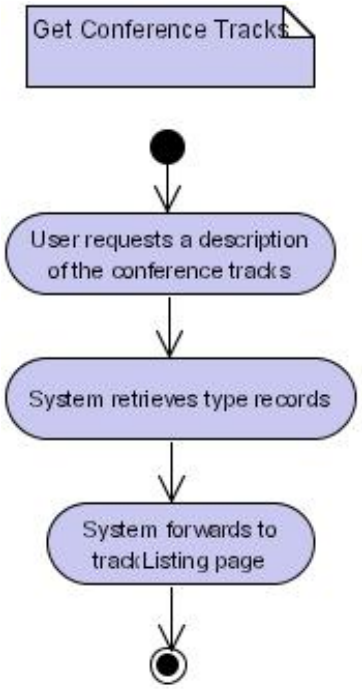

Figure 7 - Activity Diagram for "Get List of Conference Tracks" Use Case

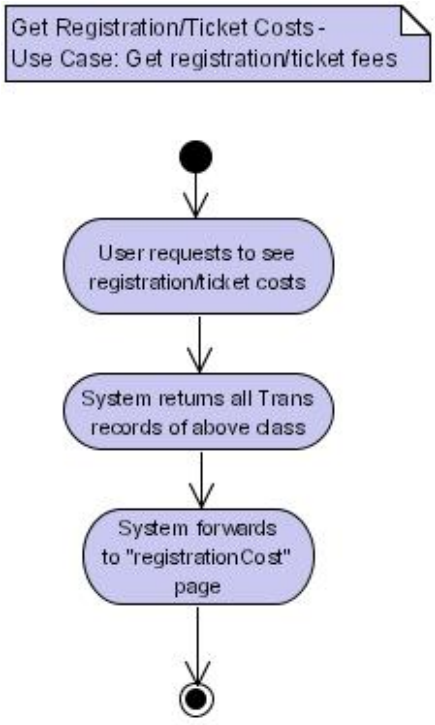

Figure 8 - Activity Diagram for "Get Registration/Ticket Fees" Use Case

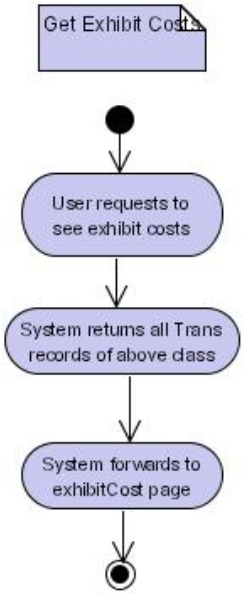

Figure 9 - Activity Diagram for "Get Exhibit Costs" Use Case

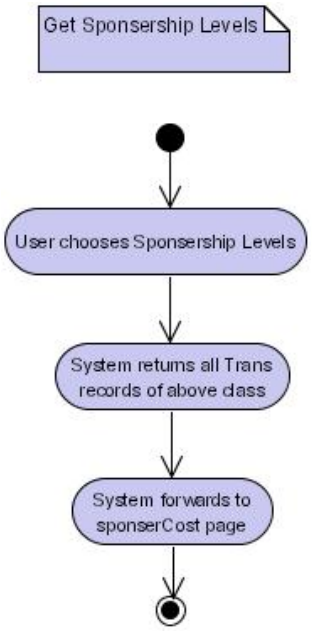

Figure 10 - Activity Diagram for "Get Sponsership Levels" Use Case

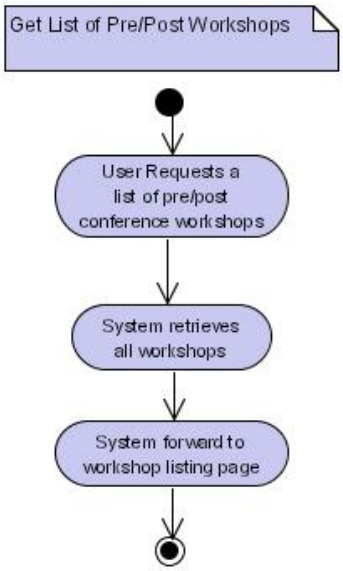

Figure 11 - Activity Diagram for "Get List of Pre/Post Workshops" Use Case

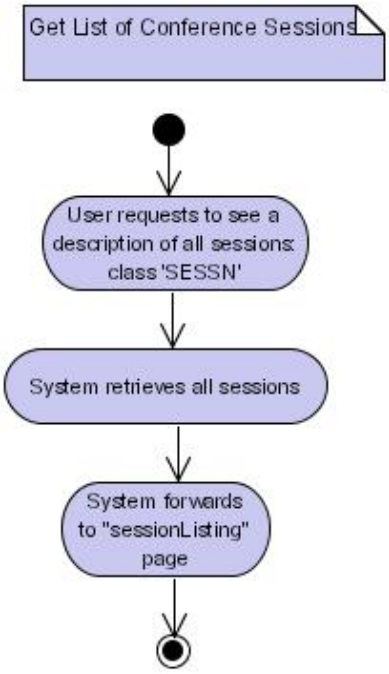

Figure 12 - Activity Diagram for "Get List of Conference Sessions" Use Case

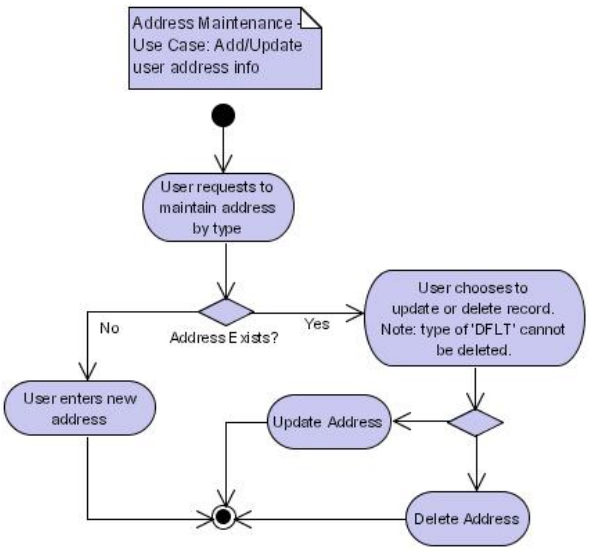

Figure 13 – Activity Diagram for "Address Maintenance" Use Case

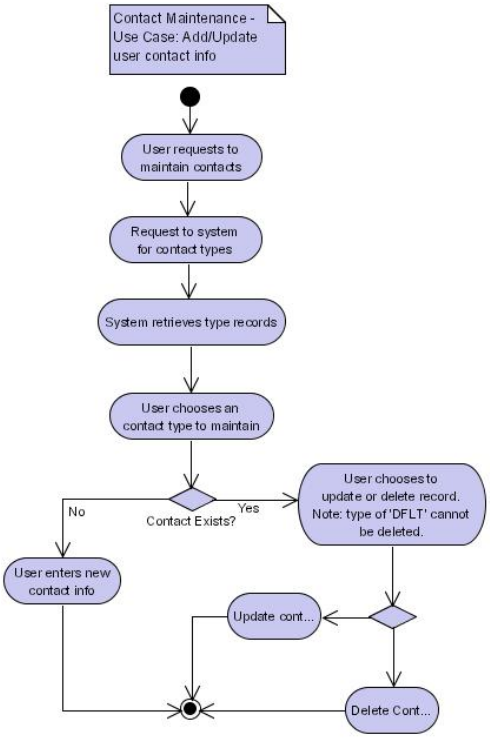

Figure 14 – Activity Diagram for "Contact Maintenance" Use Case

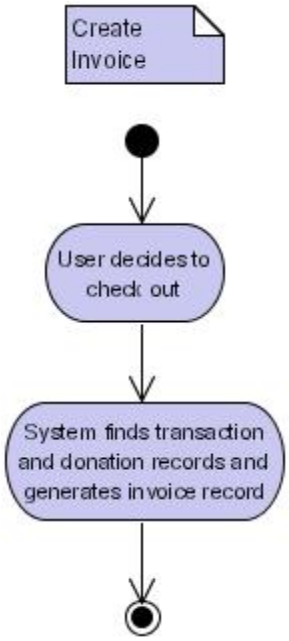

Figure 15 – Activity Diagram for "Create Invoice" Use Case

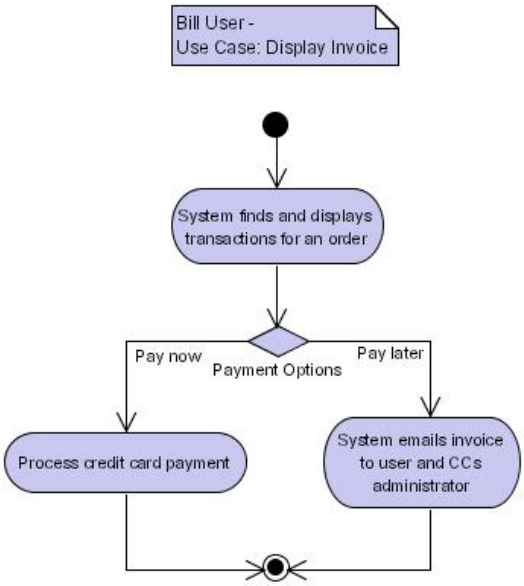

Figure 16 – Activity Diagram for "Display Invoice" Use Case

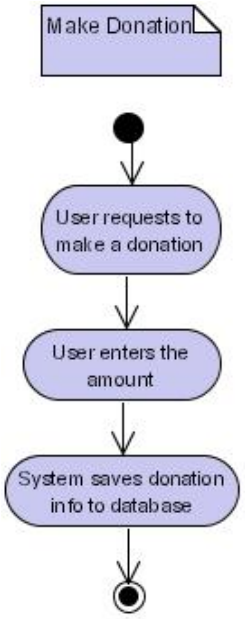

Figure 17 – Activity Diagram for "Make Donation" Use Case

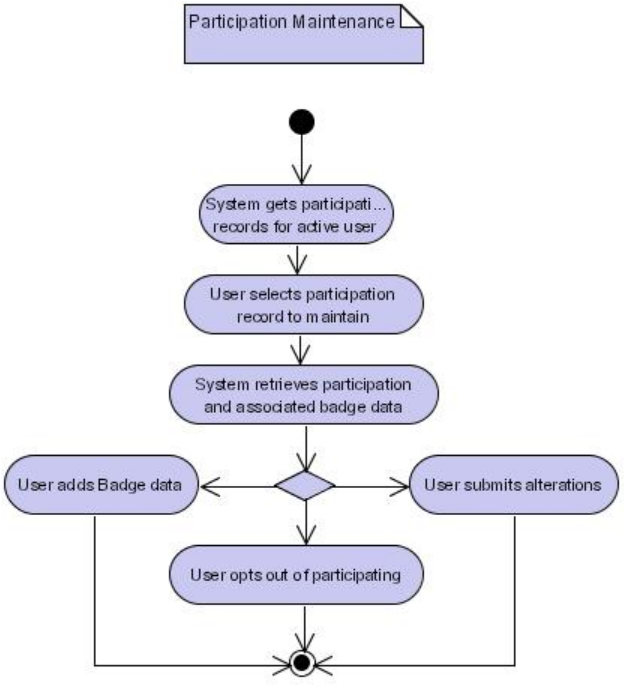

Figure 18 - Activity Diagram for "Participation Maintenance" Use Case

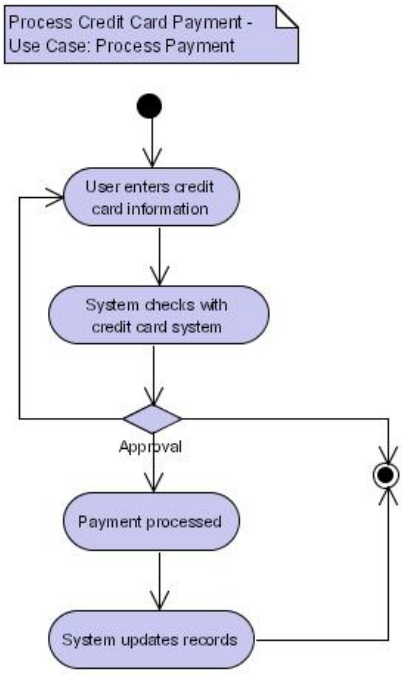

Figure 19 - Activity Diagram for "Process Credit Card Payment" Use Case

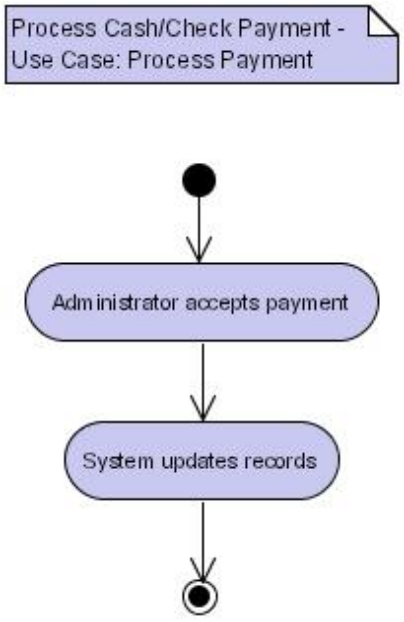

Figure 20 - Activity Diagram for "Process Cash/Check Payment" Use Case

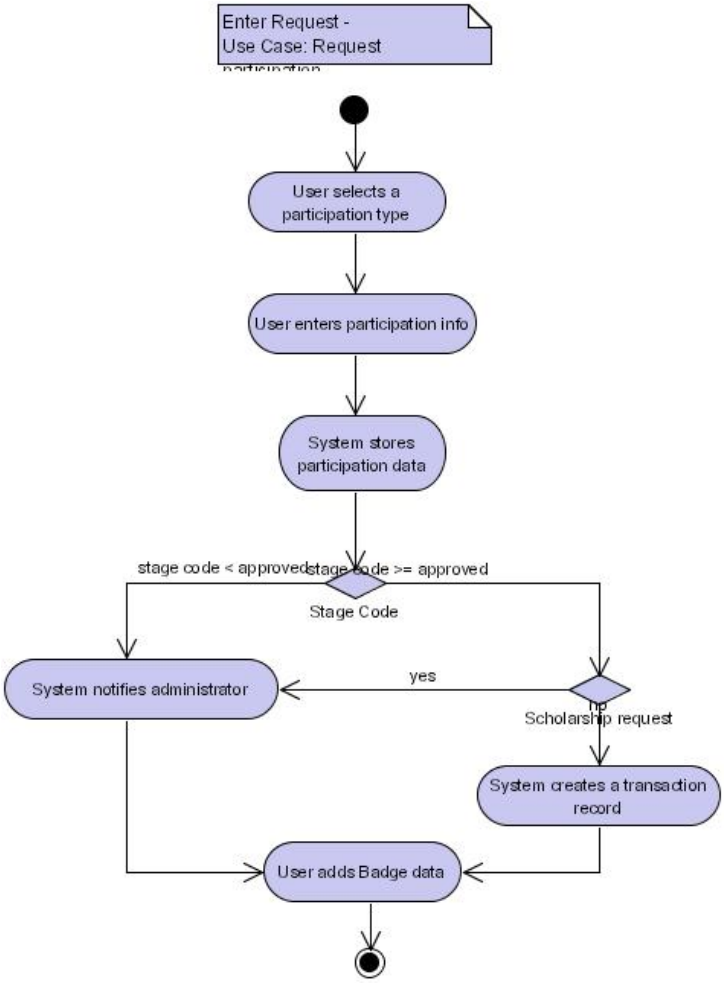

Figure 21 - Activity Diagram for "Request Participation" Use Case

APPENDIX B – DATA DIAGRAMS

Database Entity Relationship Diagram (ERD)

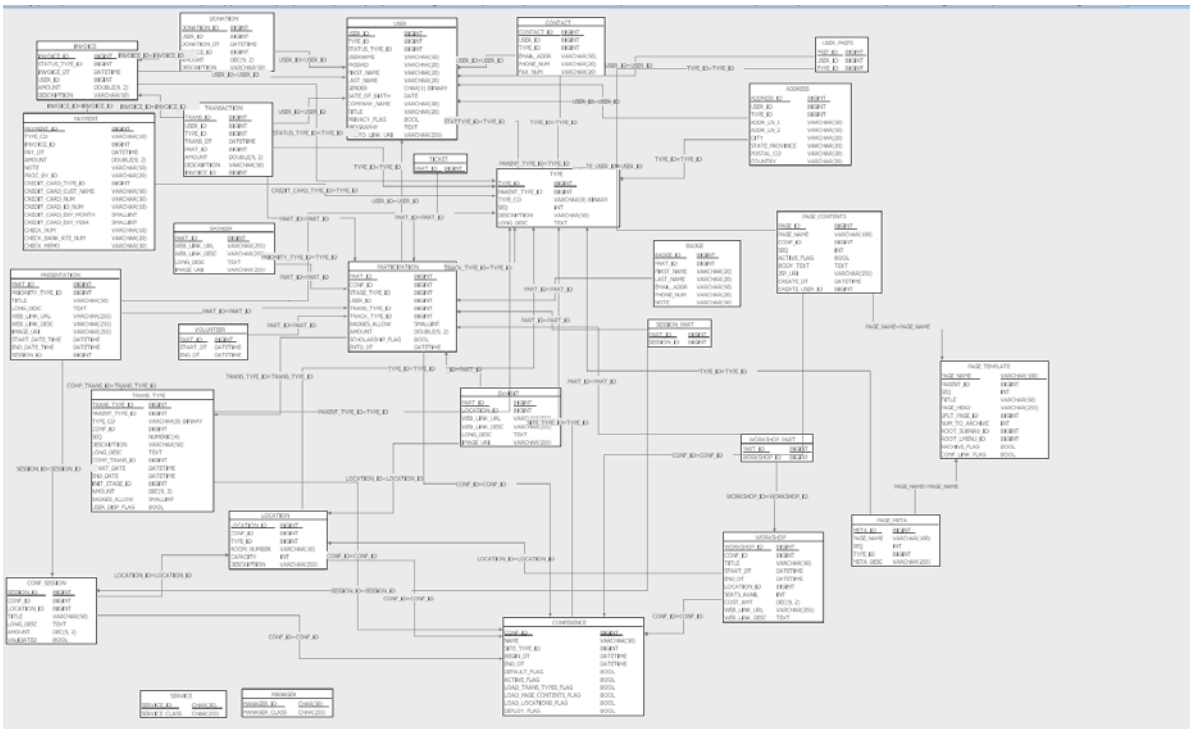

Figure 22 - Sustainable Resources Database

Static Class Diagrams of Data Model

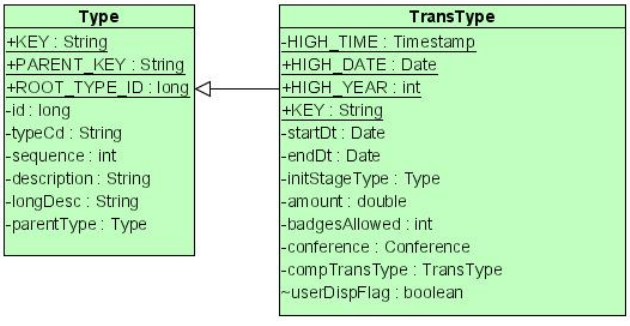

Figure 23 - Type Classes

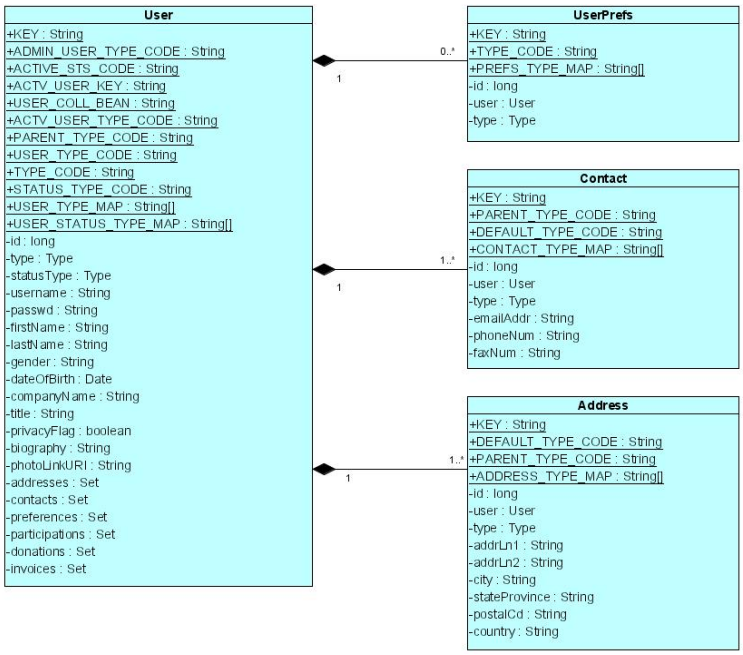

Figure 24 - User Classes

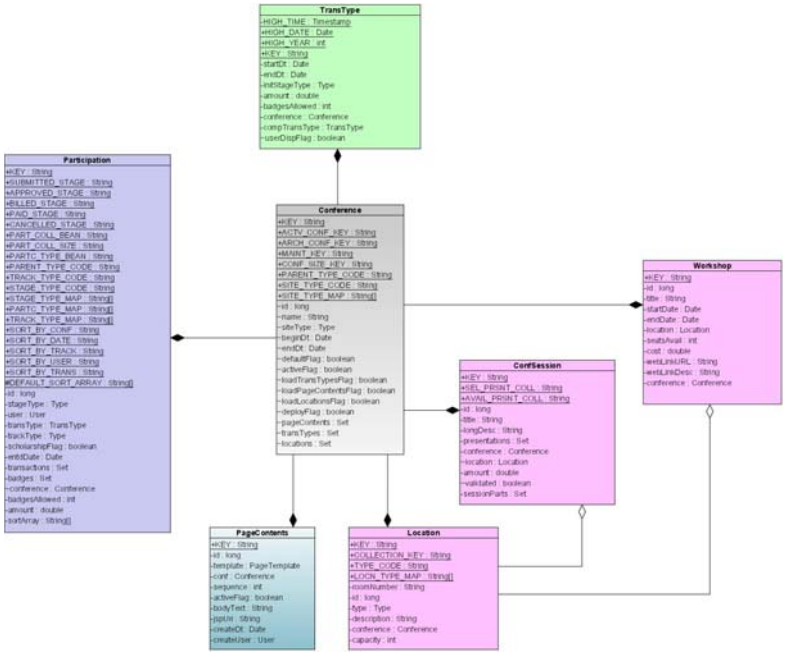

Figure 25 - Conference Classes

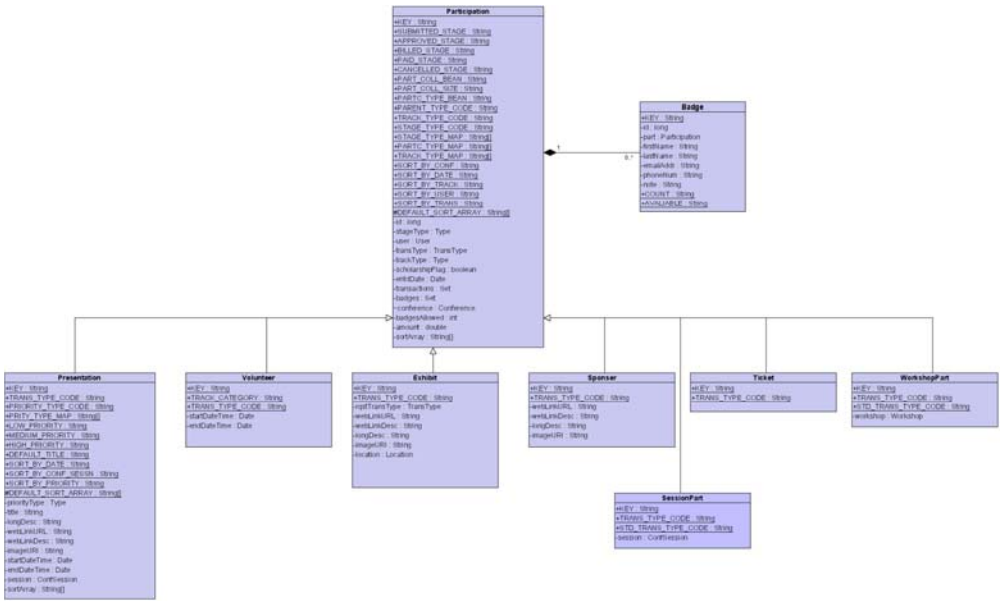

Figure 26 - Participation Classes

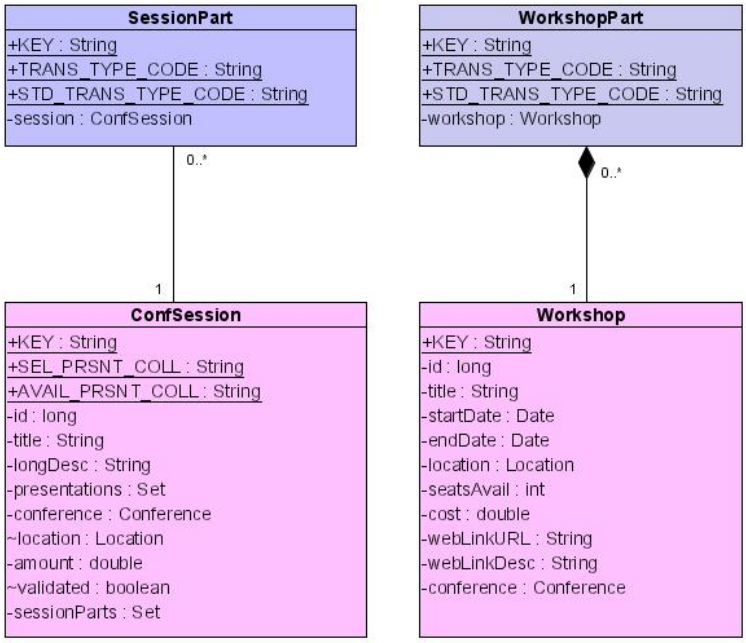

Figure 27 - Special Participation Classes

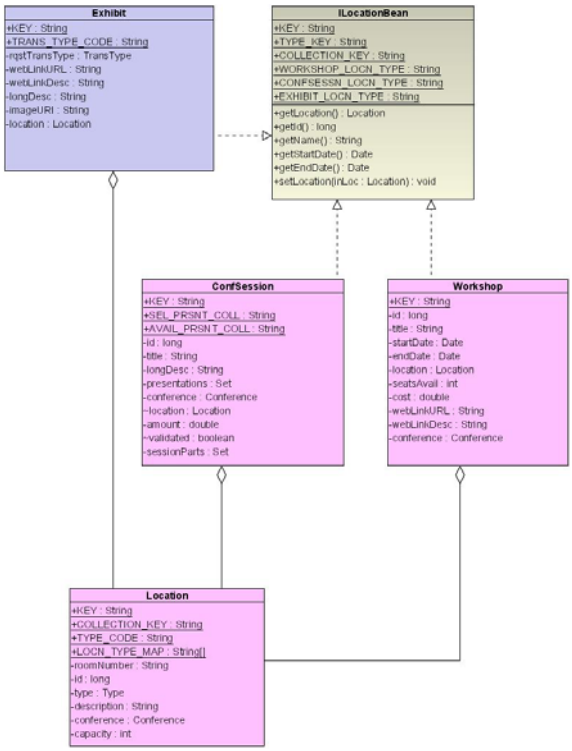

Figure 28 - Location Classes

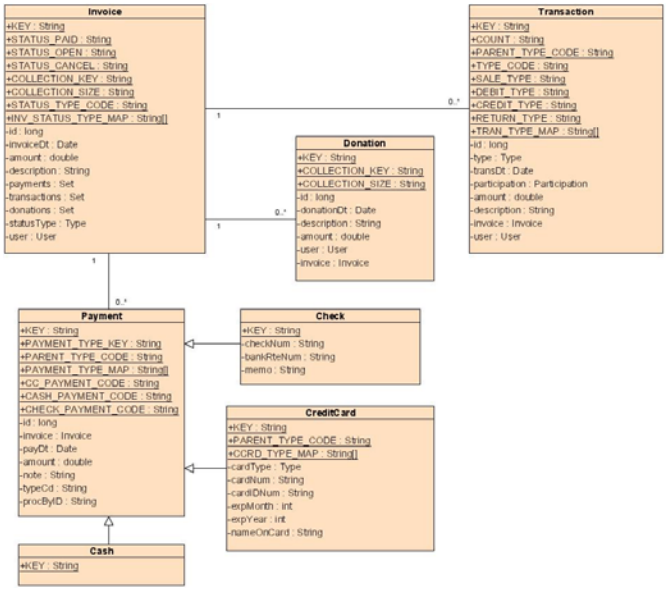

Figure 29 - Financial Classes

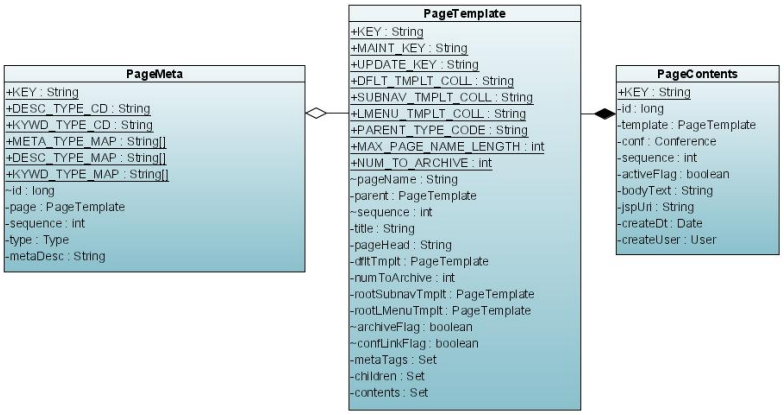

Figure 30 - Content Management Classes

APPENDIX C – SEQUENCE DIAGRAMS

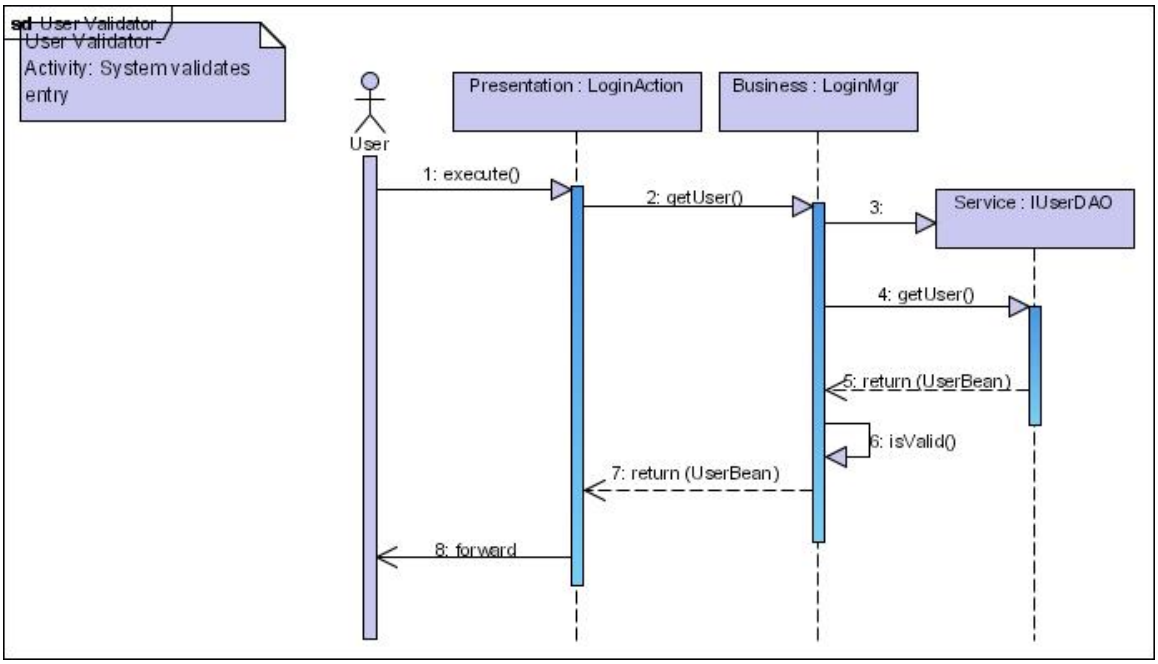

Figure 31 - User Validator

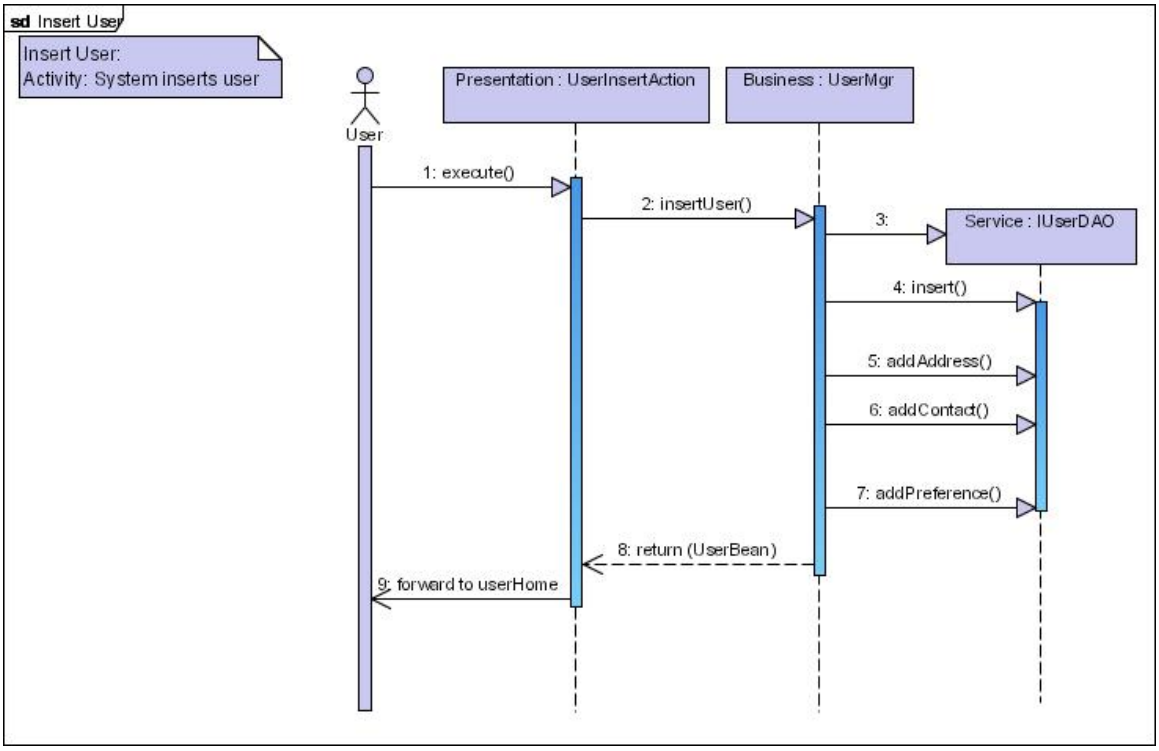

Figure 32 - Insert User

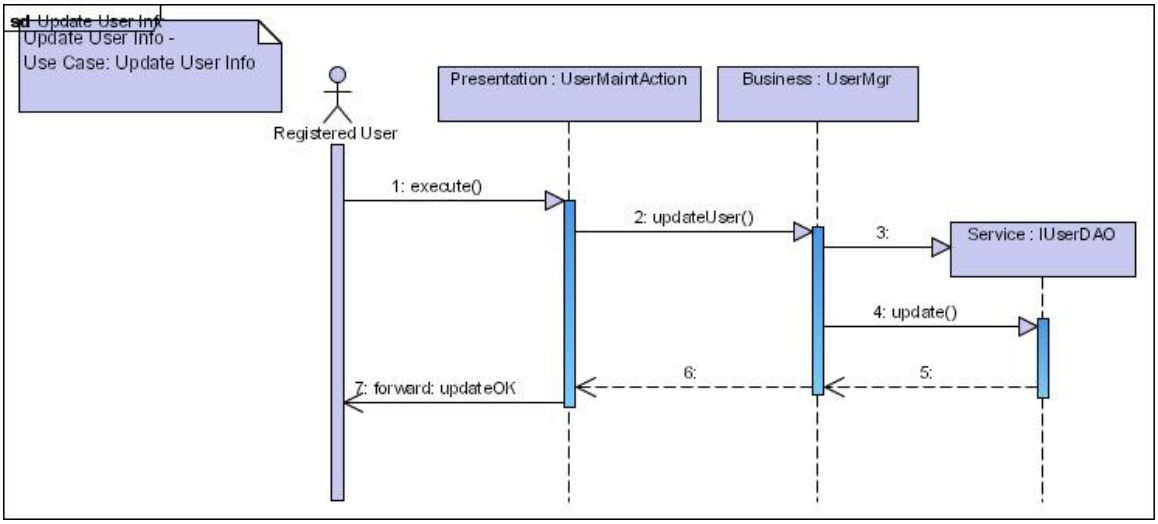

Figure 33 - Update User Information

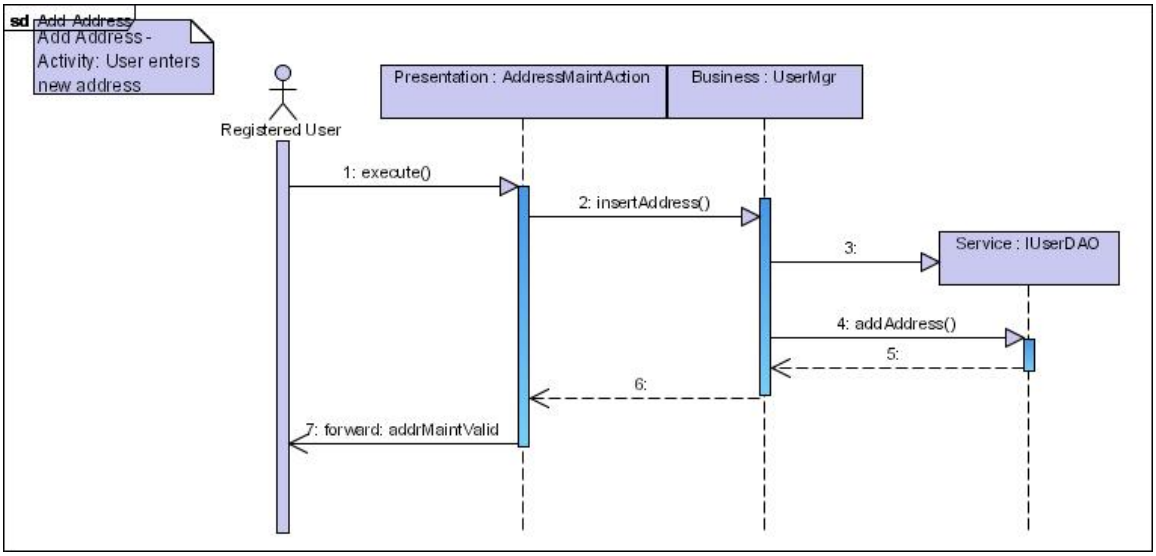

Figure 34 - Add User Address

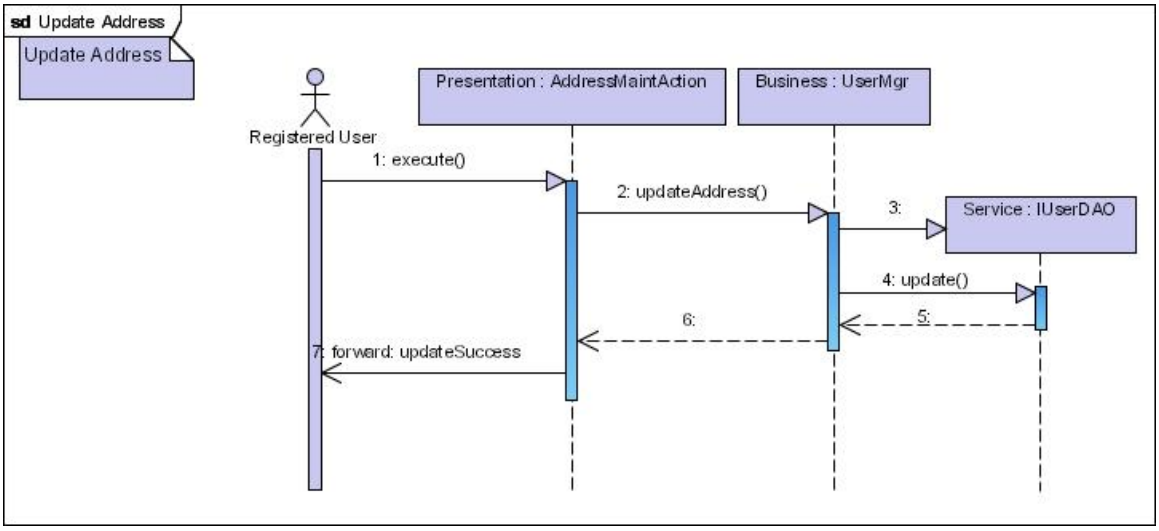

Figure 35 - Update User Address

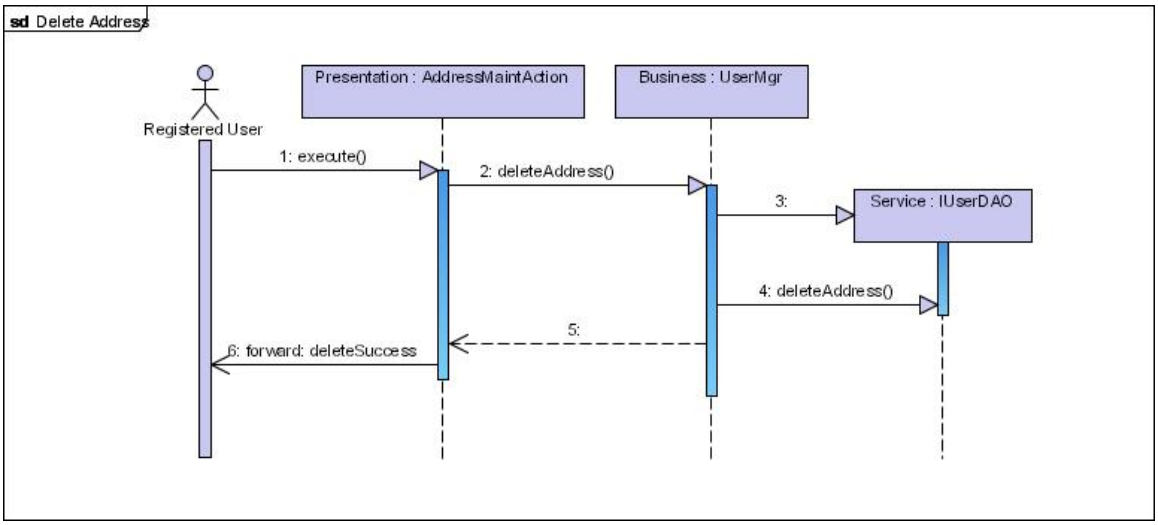

Figure 36 - Delete User Address

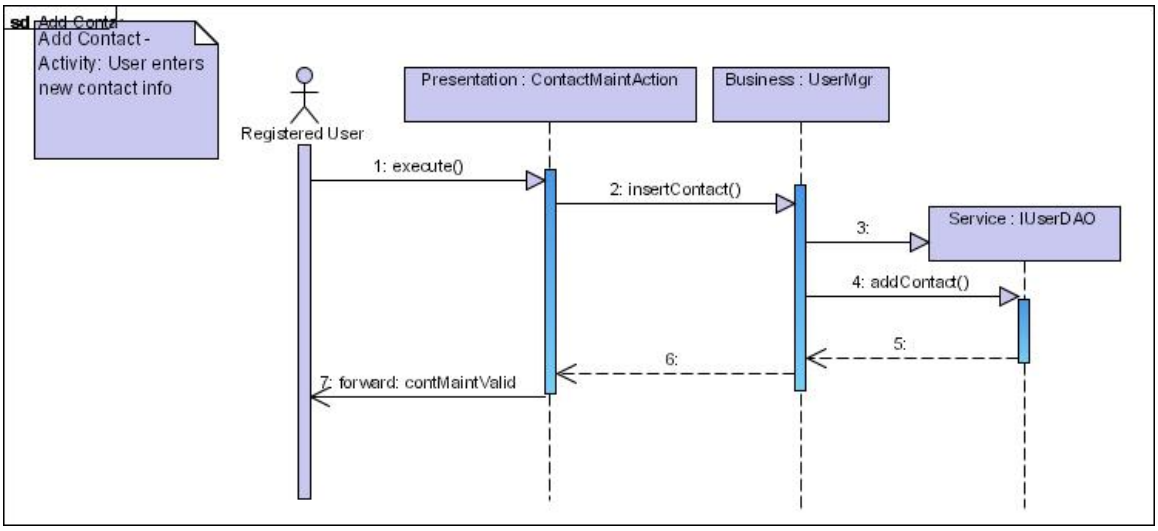

Figure 37 - Add User Contact

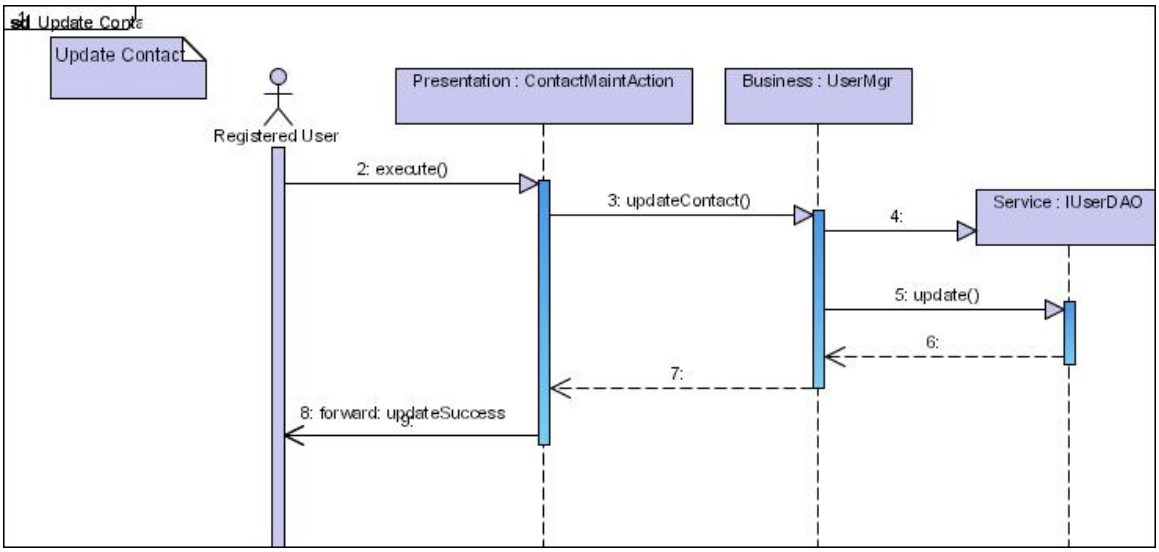

Figure 38 - Update Contact

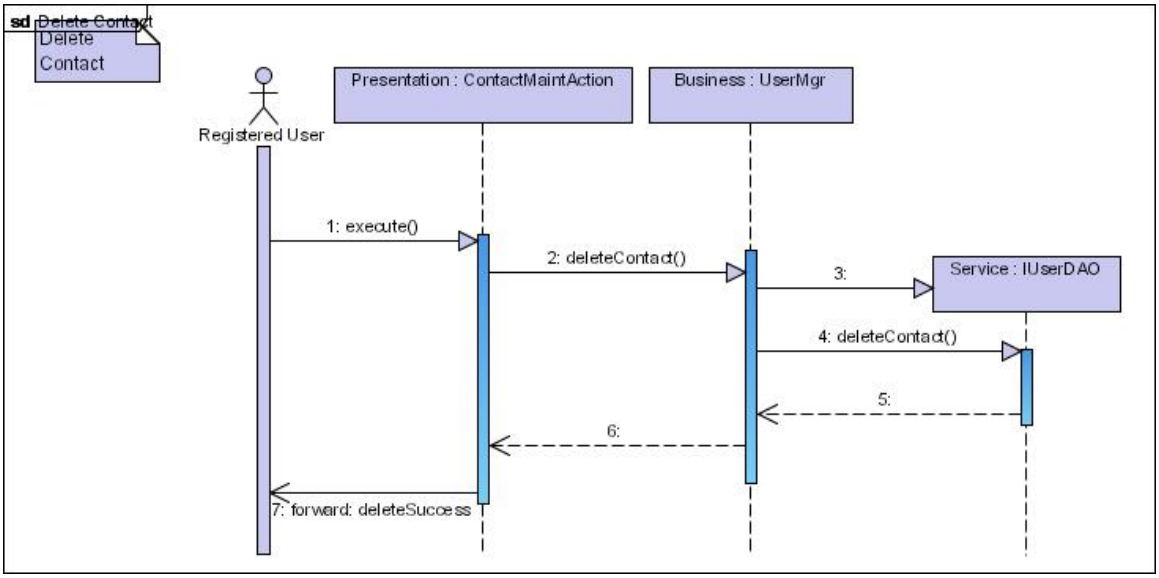

Figure 39 - Delete Contact

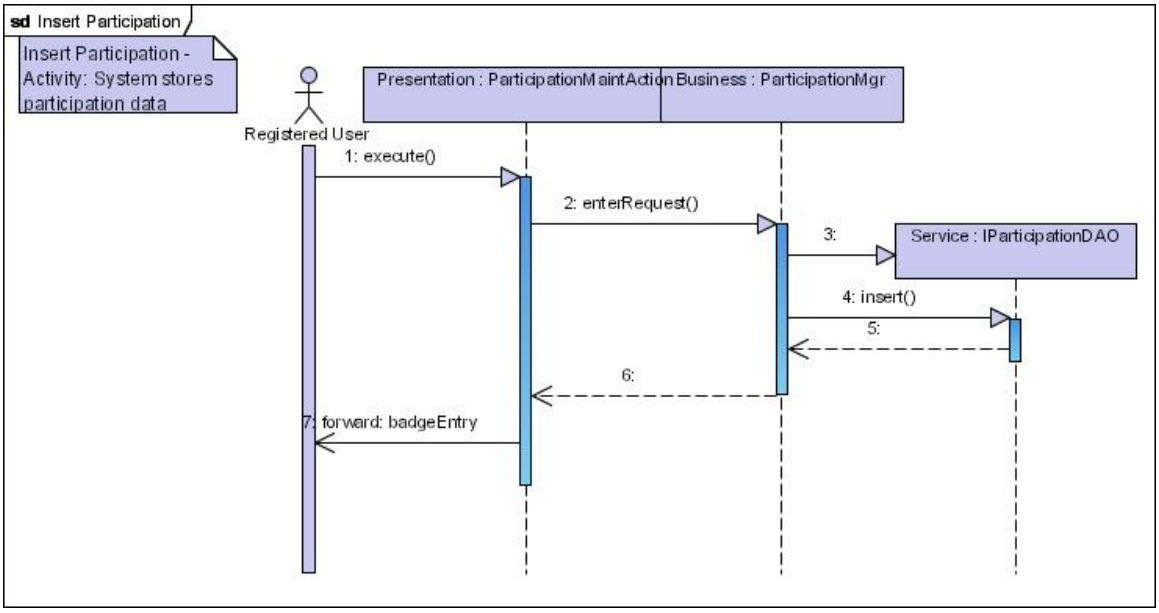

Figure 40 - Insert Participation

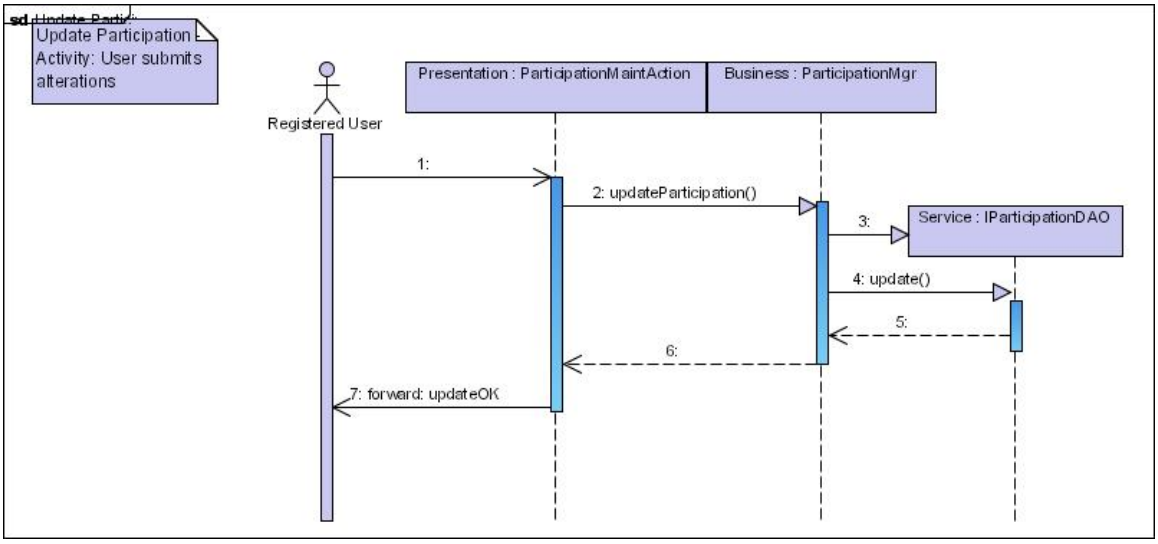

Figure 41 - Update Participation

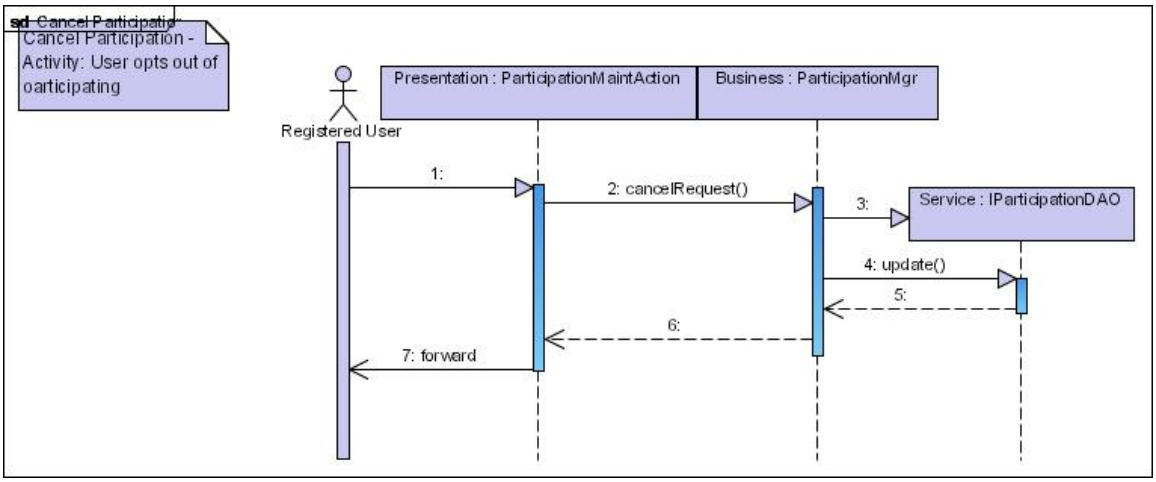

Figure 42 - Cancel Participation

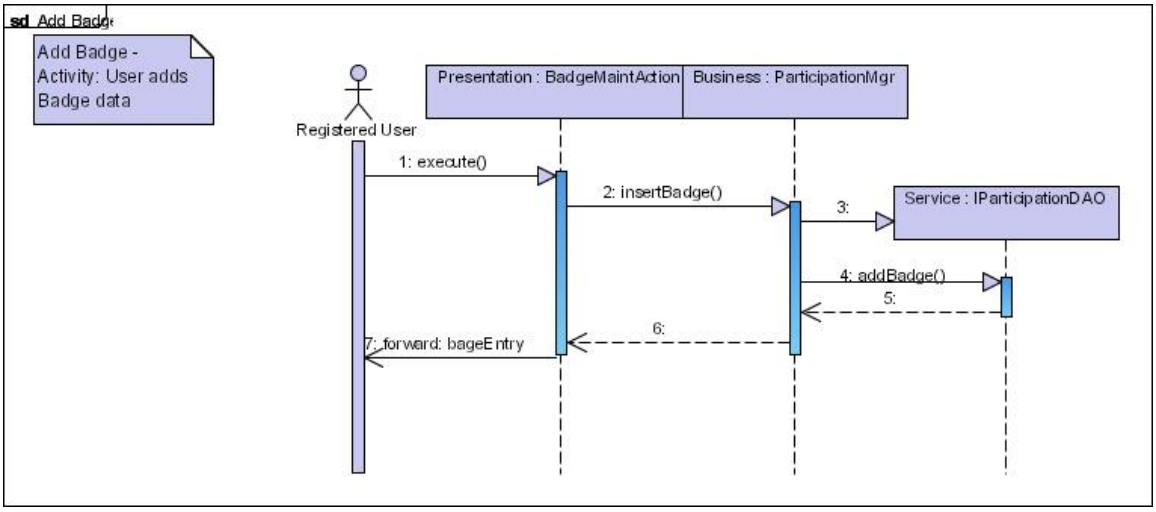

Figure 43 - Add Participation Badge

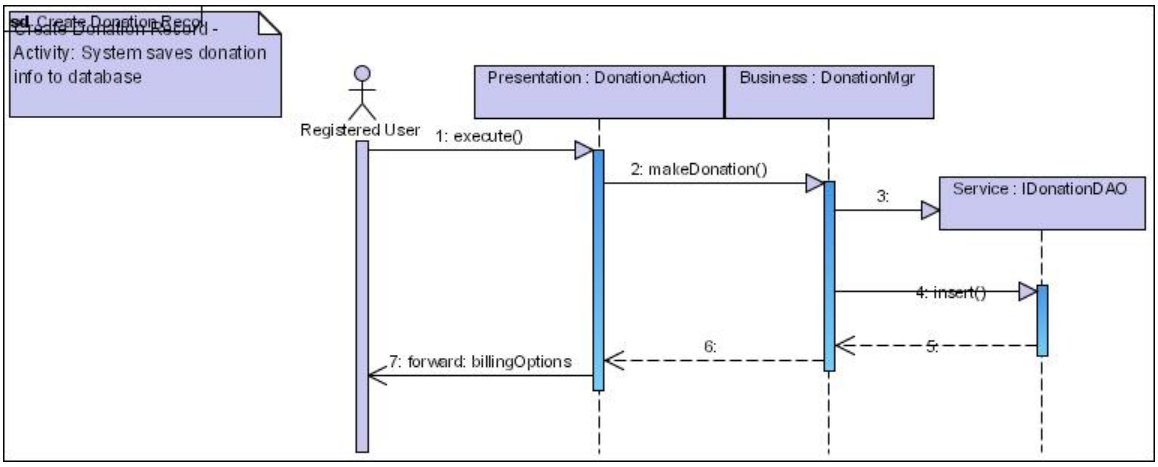

Figure 44 - Create Donation Record

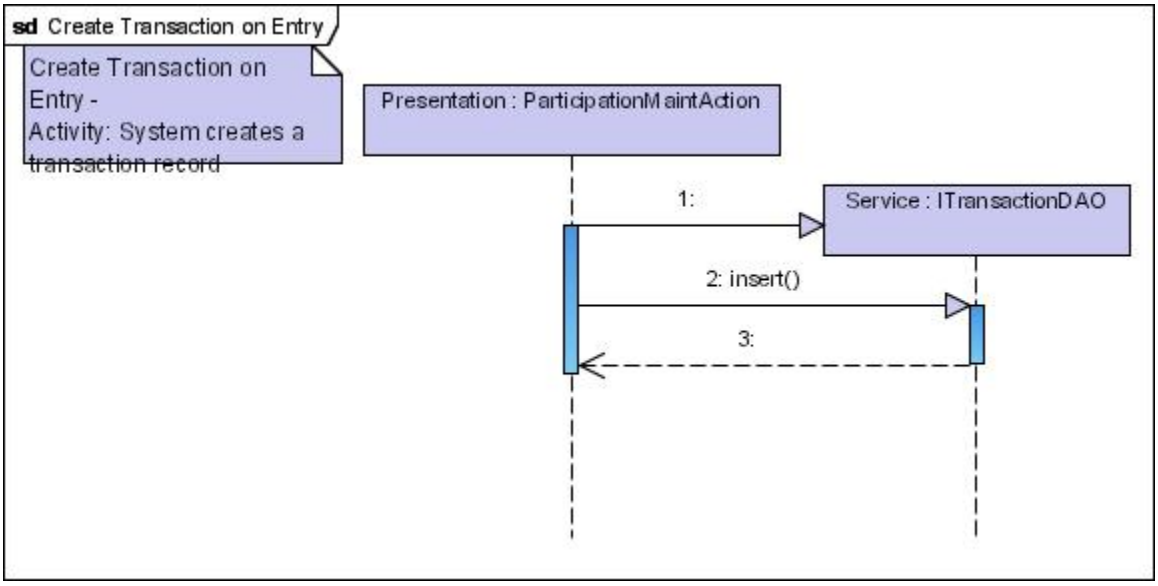

Figure 45 - Create Transaction Record

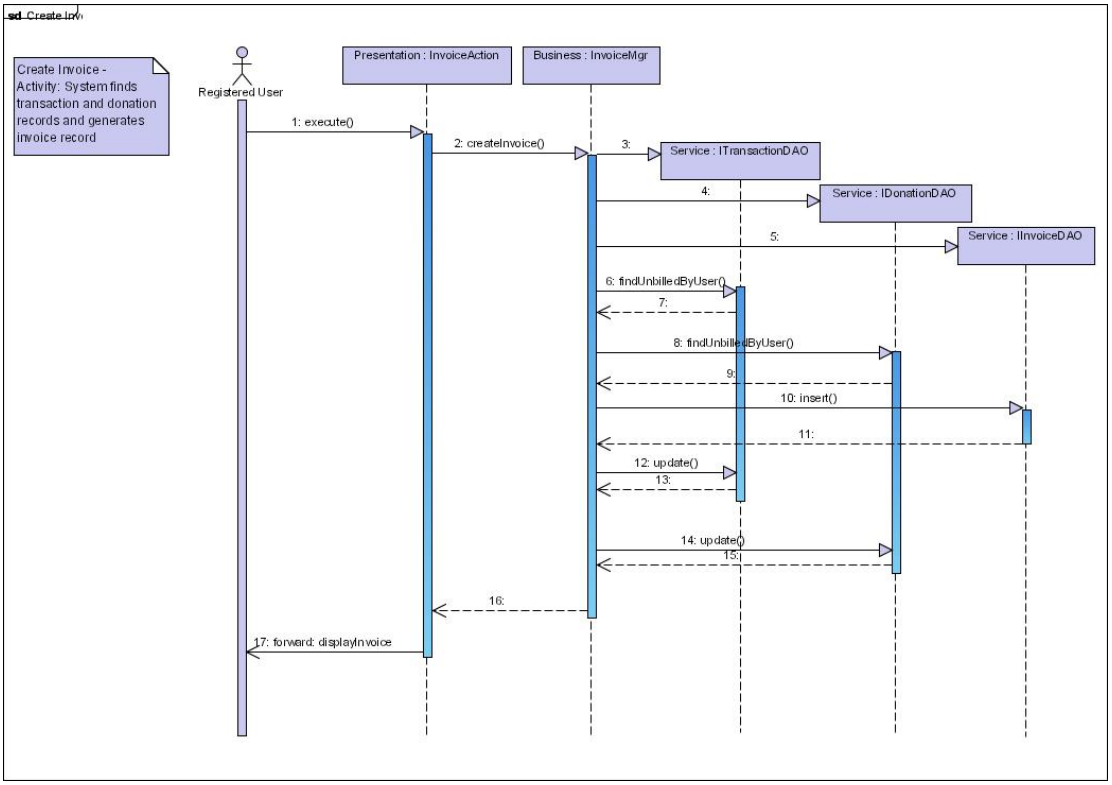

Figure 46 - Create Invoice

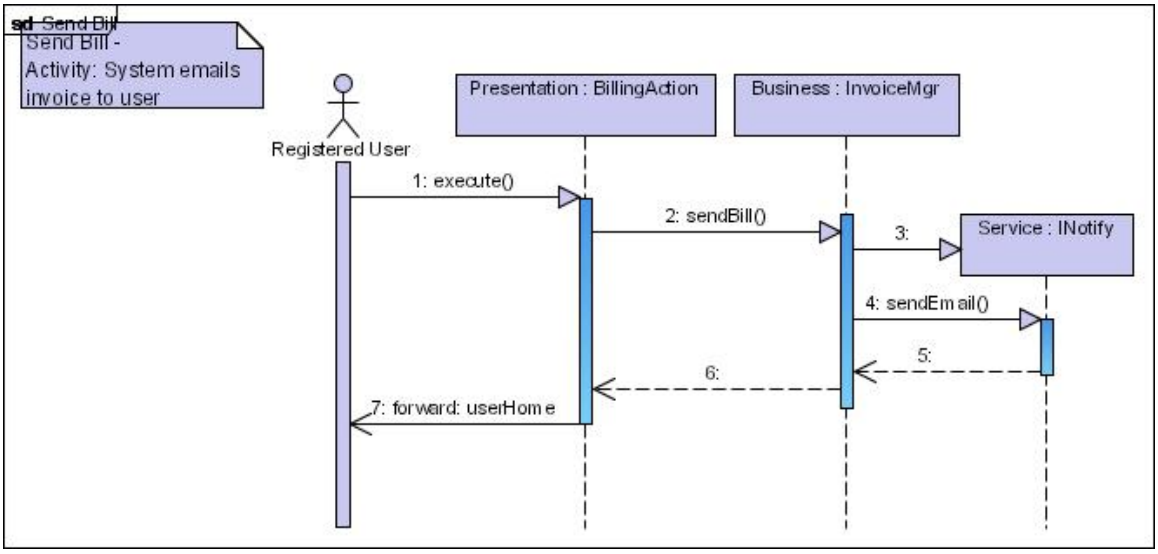

Figure 47 - Email Bill

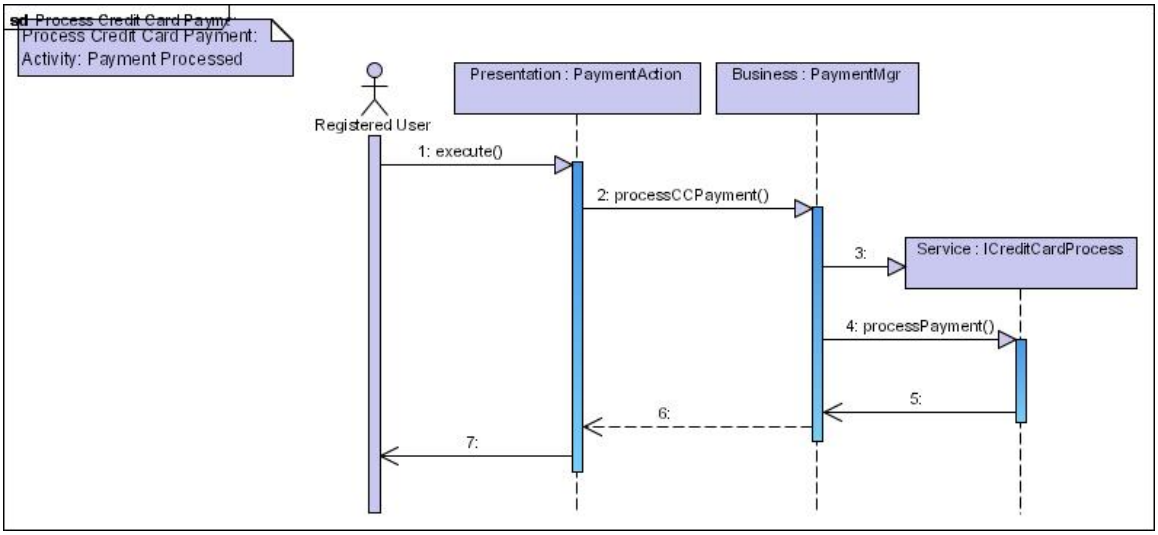

Figure 48 - Process Credit Card Payment

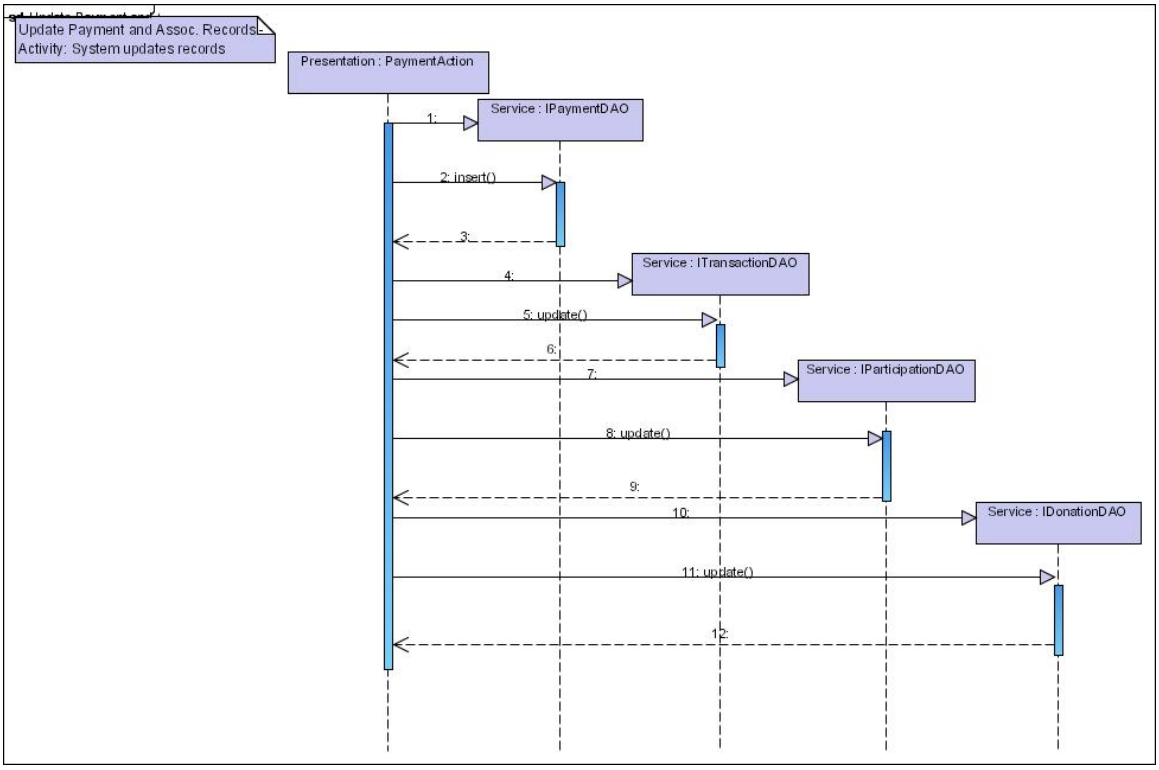

Figure 49 - Update Payment and Associated Records

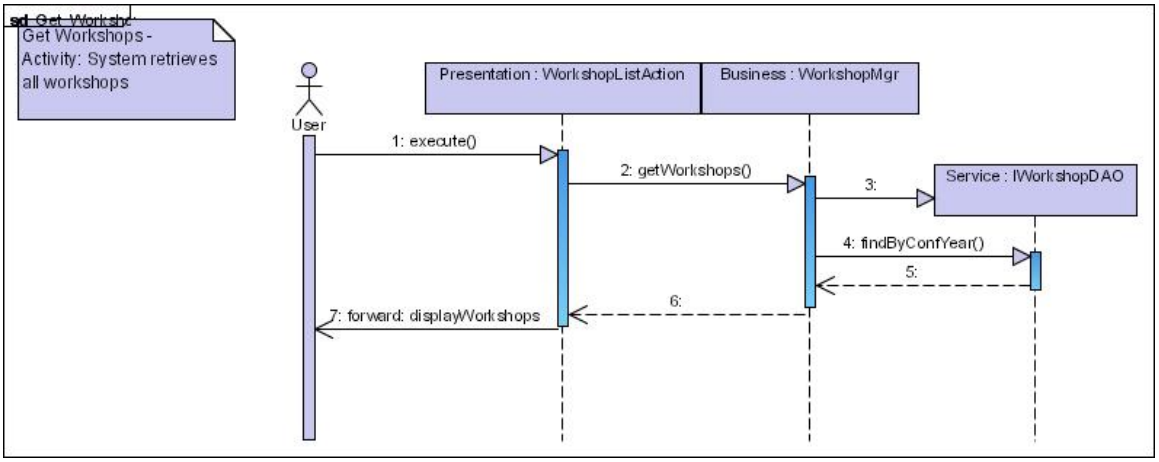

Figure 50 - Get Workshops

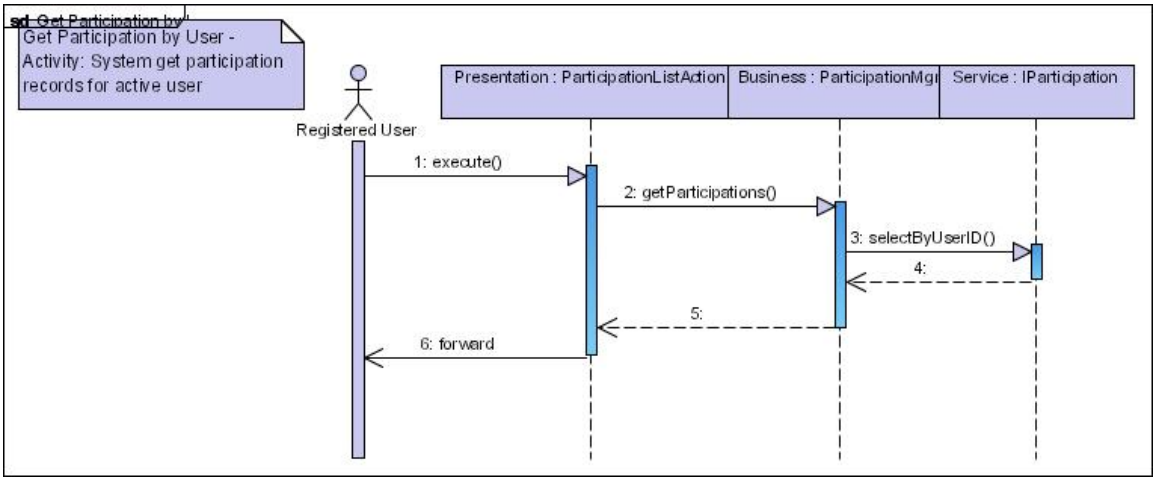

Figure 51 - Get Participations by User

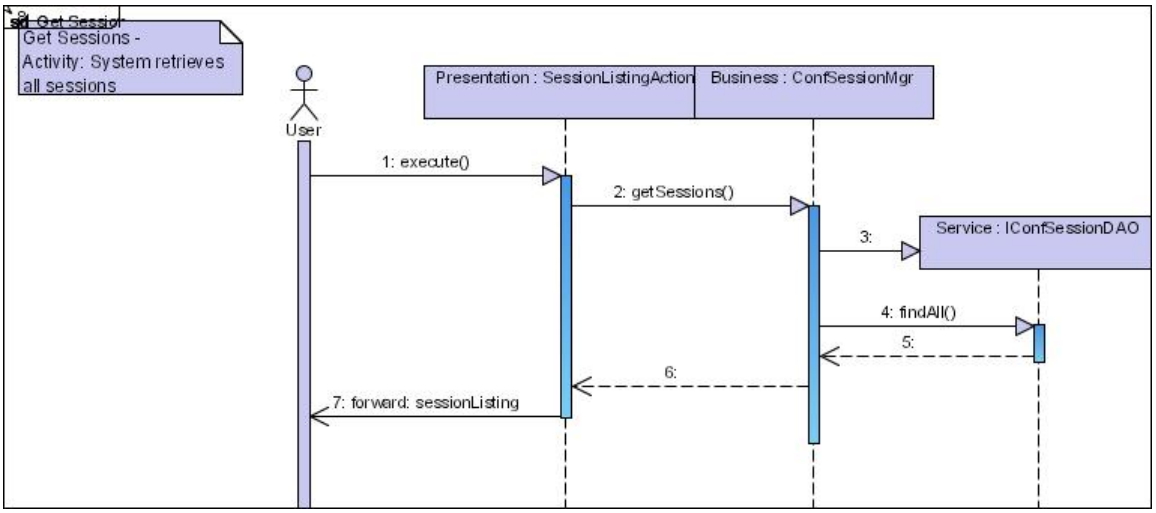

Figure 52 - Get Conference Sessions

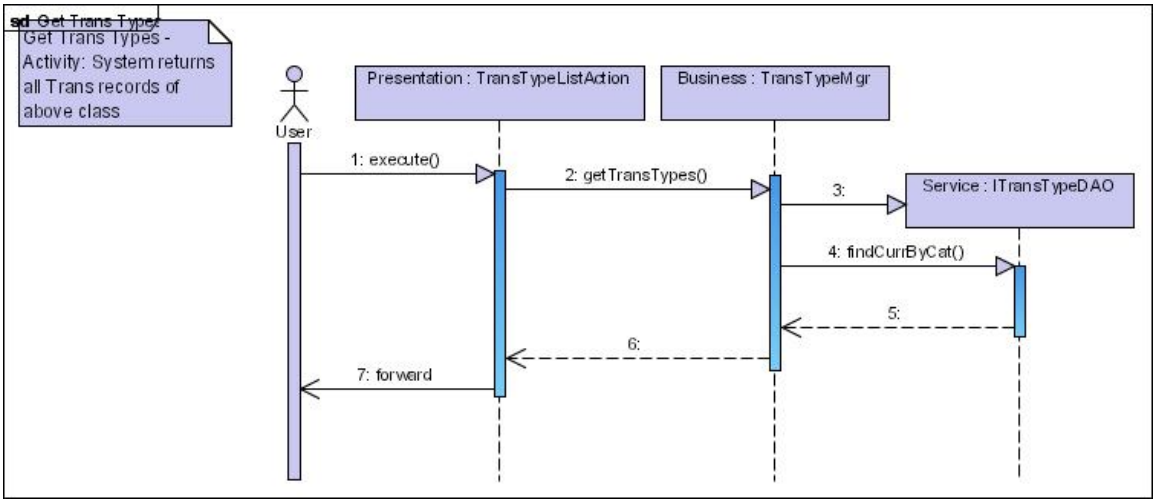

Figure 53 - Get Transaction Types

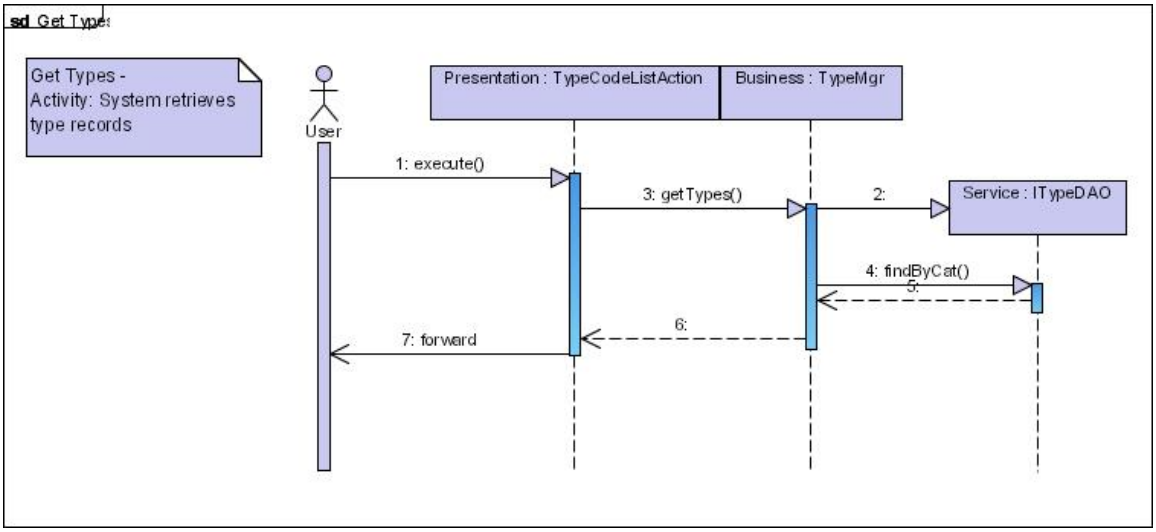

Figure 54 - Get Types

BIBLIOGRAPHY

Bauer, C., & King, G. (2005). *Hibernate in action*. Greenwich, CT: Manning Publications Co.

Goodwill, J. (2002). *Apache Jakarta-Tomcat*. Berkeley, CA: Apress.

Van Duyne D. K., Landay, J. A., & Hong, J. I. (2003). *The design of sites: Patterns, principles, and processes for crafting a customer-centered web experience*. New York: Addison-Wesley.

Musciano, C., & Kennedy, B. (2000). *HTML & XHTML: The definitive guide* (4th ed.). Sebastopol, CA: O'Reilly & Associates, Inc.

Meyer, E. A. (2004). *Cascading style sheets: The definitive guide* (2nd ed.). Sebastopol, CA: O'Reilly & Associates, Inc.

Bergsten, H. (2004). *JavaServer Pages* (3rd ed.). Sebastopol, CA: O'Reilly & Associates, Inc.

Turner, J., & Bedell, K. (2003). *Struts kick start*. Indianapolis, IN: Sams Publishing.

Richter, C. (1999). *Designing flexible object-oriented systems with UML*. Indianapolis, IN: Macmillan Technical Publishing.

Hall, M. (2002). *More servlets and JavaServer pages*. Upper Saddle River, NJ: Prentice Hall PTR.

INDEX