ISSN: 2167-1435
v29 n1965

Developing Enterprise Information Systems: Experiences of a Graduate Class

Jeffrey Holifield
jeffkholifield@gmail.com

Bart Longenecker
longeneckerb@gmail.com

David Feinstein
dfeinstein@usouthal.edu

University of South Alabama

Abstract

Teaching students how to work as a part of a team and develop enterprise information systems is a worthy and challenging goal. We describe the type of problems we attempt to solve with these projects and the structure and type of teams we form to complete the projects. We describe some of the leadership issues and concepts we teach to student leaders and followers and describe how the project is managed though Lean software development techniques. We describe the standards and procedures we use to integrate the developed systems and the use of templates to guide system development and student efforts.

Keywords: enterprise information systems development, project management, leadership, Lean UX, standards

1. INTRODUCTION

A primary goal of our graduate program is to enable students to learn, practice, and control enterprise information systems development in a team environment. Students who enter this capstone course sequence have already studied information assurance, database design and development, object oriented - systems analysis, web software development and project management. A suitable problem is identified, and short life cycle techniques are applied under student project leadership and student development to complete the project. Teams are formed to take-on project responsibilities. In the first semester, novice students are trained in lifecycle and project implementation techniques and standards by more advanced second semester students. Written methodologies presented as templates are utilized to control all project activities. Students in their second semester serve as project and/or team leaders and are responsible for the progress of their junior colleagues. Project and other team leads are thought of more as leaders and coordinators as opposed to dictators; these leaders have to earn the respect of their team members. Both team leads and project level leaders all work. Usually the leaders take-on considerably more work than do the members of their teams. Some projects are completed in a semester while other projects may span multiple semesters. While student work is done in teams, individuals volunteer to take-on specific responsibilities which they must complete. All work is evaluated initially by team leads, and

then by project leadership, and finally by the instructor. The collection of work forms the basis for a semester grade. There are usually between 20-30 students in the class, divided into project teams. While standards are utilized and are very important, the flow of the class is relatively unstructured. Students largely volunteer and self select their teams and roles as well as any specific assignments. They frequently struggle to complete their responsibilities and facilitate achievement of personal and project goals.

2. PROBLEM SCOPE

The classes of problems we take-on typically involve multiple organizational functions. Normally, solution of these problems will involve 50 to 100 tables and a similar number of tightly coupled screens. Once the scope is set, it is rarely changed during the semester. The problems are challenging and worthy of solution. Although we have worked with "live" clients in the past, we have reached the conclusion it is safer not to have a "real" client involved. This decision provides more time for instruction of junior members, and for learning implementation skills as well as team dynamics.

3. TEAM STRUCTURES AND DEVELOPMENT

The structure of an organization to a large extent drives the behavior of that organization (Jonker, 2012). We want and expect quick turnarounds, team agility, and the ability to adapt and use new techniques and technologies. To achieve these ends, we use a relatively large number of small teams with a very simple organizational structure. Most teams are composed of four to five members and are student lead.

Based on interviews, as well as previous class performance, the instructor usually selects a project manager. For the diversity of teams we have, the project manager must be a successful communicator and leader who understands and can use most of the fundamental implementation skills. He or she must also be able to help the teams function and produce the work needed to accomplish project goals. The project manager is responsible for forming the management team and insures they understand their responsibilities to the project leadership and to the other teams in the class.

Teams are usually formed within the first few days of each semester. Unless problems arise, teams remain unchanged throughout the semester. The project manager and other team leads are responsible for solving issues and restructuring team composition if necessary.

Management Team

The management team is a collection of team leads and members of the advanced team. Their activities are coordinated by the project manager. They basically serve as a staff to advise the project manager about issues that arise during the project, to participate in forming approaches that will facilitate the project to move forward, and provide expertise to individual teams as needed. The management team meets for 15-20 minutes before each full class session. The project manager is responsible for the agenda for these meetings. The management team also takes responsibility for development of a menu system for the project. Following a login screen which establishes a user ID, the main menu is evoked.

Advanced Team

The advanced team is made up of some of the most accomplished members of the class. They are experts in application development. Almost invariably, they have already been through one semester and have learned the basics of the systems development approach. The team is capable of taking on and solving very difficult problems. Some of the problems involve prototyping implementation methods; others represent problems that are considerably advanced for the class. For example, in a recent class, a member of the advanced team reverse engineered the java script .dll, fixed numerous errors, and rebuilt the .dll successfully. Also, when team members cannot solve a difficult issue, members of the advanced team may be consulted for advice.

Systems Analysis and Design Teams

During the initial phase of problem scoping and definition, requirements development, work flow generation, preliminary database and screen design, the class works together with the instructor to develop an approach to the problem. The teams assist by completing templates (See section 6), which embrace the developing design. As the system begins to factor into component sub-systems, the teams identify the detailed database tables and screen

layouts for their aspects of the project. All of this material is reviewed in a sequence of structured walkthroughs involving the whole class.

Database Team

The database team is a small team utilizing the most talented database experts in the class who are interested in developing the database and sample data. This team is one of the most important groups because not only does the implementation have to be done well, it must be done quickly. We build databases by developing scripts. As changes to the database are required, the script must be revised and sent to all class members. Teams may develop and test new ideas in the database, such as stored procedures needed by their screens, however, these ideas must be immediately communicated with the database team. The database team must turn around changes to the class database within the same day, and get the revised script back to the teams. If the database team detects errors they immediately contact the submitting members to review the issues. The database is so critical to the success of the project, that delays must be avoided if at all possible. Once changes are made, the new database script is posted on the class website and the class immediately put the new script into use. If old copies of the database are allowed to persist, field names may have changed, new tables may not be referenced, and inconsistency in sample data can develop very rapidly. This makes future integration of team screens very difficult. During each class, the management team is tasked with verifying that all class members are using the revised materials.

We utilize a parameter passing mechanism called a session variable table (SVtable). The database team is responsible for maintenance of SVtable. Each variable that must be communicated between application pages is stored in SVtable. Both utilizing Inline Frame (IFrame) functions and moving parameters into and from SVtable are managed by our Dynamic Link Library (.dll) (see appendix A).

Application Development Teams

The application development teams are responsible for implementation of a set of screens which will comprise the completed application. Each team member takes on a set of screens to implement. Each screen is a web page that must use the cascading style sheets (CSS) developed by the advanced team for the project. Applications are developed in an N-Tier approach learned in earlier courses. However, each page is invoked through an IFrame technique under control of our .dll. Team members are instructed in the IFrame techniques and the use of CSS by videos made available on a class web site.

Integration and QC Team

The integration team is another small but important team composed of experts with the ability to spot and recommend correction of possible errors in pages to be integrated. So long as the current database is used, the right connection string is used, and the application "works", integration should be minimal. To minimize integration problems, we developed a set of Digital Rules of Engagement (see section 5 and appendix D), which set standards in several categories such as naming conventions, connection strings, languages, and cascading style sheets, to name a few. It is the responsibility of the integration team to detect and document errors, not fix them. Defective pages are returned to the teams, perhaps with commentary. It is important for the integration team to turn all submissions around in the same day in order to keep the project flowing. Revised pages are submitted via email, and error reports are generated within the same day.

We have attempted to use check-in/check-out systems with limited success—indeed we have lost a lot of time with software that does not work. Instead, team leads are asked to coordinate page access to prevent different team members or different teams from simultaneously updating the same page.

Facilities Management Team

The facilities management team is responsible for our class servers and training modules. Likewise, the team is responsible for keeping current software running and configured. The team interacts with the university computer services center to ensure appropriate network configuration and connection to enable off-campus access for database and integration updates. We utilize remote access to work directly on any of our servers.

4. PROJECT LEADERSHIP AND MANAGEMENT

As stated earlier, the instructor selects the Project Manager. Leadership is perhaps the most distinguishing skill for the project manager. The project manager need not be a charismatic leader, but must understand the principles illustrated well by Raccoon (2004) and by Maxwell (2011). In addition to impeccable ethics, "leaders must use their influence to promote people and purposes" and must "treat followers as people, evoking their best, not competing with them, and not abusing authority." (Raccoon, 2004). While positional authority, that authority which is inherent to the position a leader holds, may exist in the military or even academia, a peer leader in our class has little inherent authority. The student leader must work to earn the respect of the teams.

Brewer (2005), contends that while some leadership skills can be taught and retained, some innate characteristics, such as integrity, creativity, and the ability to inspire others, are needed in leaders because it is much more difficult to change personality than to learn and retain new skills. There is no question the leader must be trustworthy. Likewise, the leader must express a clear vision of the project goals. The project leader serves as a catalyst, and indirectly oversees the project steps or phases (Raccoon, 2004). That is, the leader cannot force anyone to volunteer—team members make up their own minds—team members set their own goals. So it is the job of the leader to help build the teams, and to attempt to influence those who become followers. The project manager must provide the support and encouragement necessary for the member's success, the individual teams success, and thereby the project success. Indeed, effective leaders will help their followers to become more advanced because of the leader's effort. Leaders must be involved as well as being physically present—they do not need to fill the commander role. Leaders have to set an example in all respects. As one of our former professors said a "leader is the first among equals" (Zenkert, 1990).

Maslow's classic theory of human motivation (Maslow, 1943) leads us to believe that there are processes we can use to motivate team members. In our groups, physiological and safety needs are assumed to be meet. Team membership, especially those teams that are self-formed, can be used to meet the belonging needs. The project manager and team leaders can use esteem needs to motivate members. The instructor encourages this by giving examples of past students who had accomplished great things after leaving school and building a career.

Yukl, (1994), proposed a multiple linkage model of leadership that showed how leader behavior and team member variables such as effort, task skills, organization and resources impacted performance of the team. For instance, we placed special emphasis on training to ensure team members could function at required levels. Members of the management team designed and delivered presentations on Lean User Experience (Lean UX), CSS, Language Integrated Queries (LINQ), and Asynchronous JavaScript (AJAX). These techniques and technologies were then applied by teams and team members in developing and producing the system.

Rear Admiral Grace Hopper (Jone, 2012) said that you lead people, and manage things. An integral part of each project is to give students the opportunity to learn and practice leadership and management skills. To accomplish our goals each semester, our project leadership and management team must be responsive to achieving system development goals and be able to do this with an immature yet developing project staff. Second semester members are far more advanced that novices, and have the responsibility to develop skills in new people. Yet, the "advanced staff" is also novice at leadership for the most part. What this means is that the principles of leadership (Raccoon, 2004; Maxwell, 2012; Covey, 1994) must be explored both by project management as well as by the instructor. Our observation is that as the project manager sets the example, and as team leads are empowered to accept leadership responsibility. Since the project manager is chosen as the most knowledgeable in project leadership as well as in the fundamental skills, this process usually works fairly well. Points of stress occur when due dates slip, significant unexpected technical difficulties are encountered, or when mistakes in design approach are made. Under these conditions of stress it is very easy to become reactive, and become either visibly upset or angry. These are moments for review by the instructor, project manager, and perhaps team leads. At his first plenary keynote address of the Association for Information Systems, Wes Churchmen (1997)

told the audience that one of the guiding principles of information systems required that professionals practice "love and kindness". Encouragement and support of team members, team leads, and others involved in the project goes a long way towards success.

After looking at several aspects of extreme programming and agile software development, we settled on an approach more agile in nature and have made use of the concept of the Kanban board (Kruchten, 2011). We have attempted to implement a version of Lean UX (Cyrillo, 2011). To be sure, project goals are important, however at the level of development of our students, considerable effort in developing elaborate work break-down structures are usually futile since most students do not have the experience to make appropriate estimates utilizing new technology they have not used; unexpected situations develop, making detailed planning much less important. What is more important is to have or develop a clear sense of steps to accomplish project goals utilizing rigorous standards to accomplish them. We use a training video for Lean UX and Kanban to acquaint all with the methods, and utilize a home-grown set of forms and reports to manage the process (see section 6).

In the most recent iteration of the class we developed a database with a web interface to replicate a Kanban board. We found that it did not work as well as we had hoped. The advantage of a Kanban board is that it is visual. Everyone can see what is in progress and who is working on it. Because we did not have a lab dedicated solely to our class, we could not build a physical Kanban board. We found the database version of great use in tracking issues and solutions, but problematic in actually providing the teams a visual way to track progress. Each week we generated reports to give to the team leads to verify that issues had been resolved.

As with any project, information flow from the leadership and customer to the teams and back is critical. The US Army uses a common sense approach to command and control. When issuing Operational Orders, two levels of command are briefed at the same time (US Army, 2000). For example, company commanders and their platoon commanders might receive the same briefing. This insures that everyone in the chain understands the commander's intent; that is what is expected of them and their units. In dealing with the teams, we took a similar approach. With the instructor acting as the customer, screens under development were reviewed by both the project manager and the team leaders, and often with the entire team. This insured that more than one person understood what the customer needed. In addition to the digital Kanban system, we often printed screenshots of the screens being developed and made notes directly on them to show changes or additions.

Perhaps the most effective management tool we used were a set of standards that served as a basis for training our teams and easing the inevitable problems of project integration.

5. STANDARDS

The first golden rule of human computer interface (HCI) is consistency (Shneiderman & Plaisant, 2010). Several of our standards were attempts to insure consistency. As with our management approach to structure, we attempt to build consistency in from the ground up. The .dll we use to implement IFrames is designed to produce a consistent user experience. The IFrames provide a useful mechanism to reach another HCI goal, that of coordinating multiple windows (Shneiderman & Plaisant, 2010). The IFrames help users understand where they came from in the system and returns them to the correct spot after they have saved or closed a form.

IFrames are the equivalent of a window. They may be positioned anywhere on the screen. They have a height and width, and are positioned down from the top of the screen, and to the right of the left side; within the window, a page may be displayed. The first design issue is how big to make the page to fully contain what is to be displayed. The second issue is where to position the page showing through the IFrame. Usually, subsequent pages are displayed to the right and below to enable anchoring (see Appendix F). Also, subsequent pages are usually displayed on top of the previous page. This is accomplished by setting the z-index appropriately. The JavaScript IFrame Movement Dynamic Link Library (JSIM.dll) enables the frame to appear to emerge to a target position, and later retract to the starting position. The speed of this motion may be set parametrically. The frame also may be repositioned by the end-user. These actions require the use of frame

motion controls also in JSIM.dll (see appendix for a brief discussion of the JSIM.dll functions).

We developed a Cascading Style Sheet (CSS) file customized to the project and the IFrame.dll. We crafted a CSS reference that described the CSS and showed where and how to use it. This guide defines the different classes in the CSS, and using code samples, shows the members of the teams how to use the CSS within their portion of the project. The use of the CSS and guide helped insure internal consistency in the project.

We also developed a single page guide for HCI to help the teams insure consistency across the system. This document covers many areas such as the size of fields, the size of the IFrames, and the format of date fields.

Finally, we developed a document called Digital Rules of Engagement (DROE). This document is used by the teams to help prevent problems during integration, and to insure that future classes could determine their intent as they try to use or modify the code within the project. Any large software development effort will present integration issues that if not resolved can doom the project before it begins and send it to an "integration hell" (Kim, Park, Yun, & Lee, 2008). The DROE is an effort to identify and resolve these issues before they impact the development cycle. Since new issues arise as in the course of the project, the DROE is intended to be a living document. The intent is to pass it to future project teams so that they can benefit from our experiences and add to them. Our DROE covers naming conventions of objects, languages to be used, connection string definition, database updating and use, and comments within code, among others. With many different teams working on small part of a larger project, it is critical that every team understand and use the DROE and the standards it defines to avoid a train wreck when we try to integrate the system.

6. TEMPLATES

Over a period of several years, we have developed a set of integrated templates that not only teach a systematic development process, but guide teams to document each step of the process. This set of templates start with a Scope Document which captures the primary focus of the project in a short concise document. It includes the purpose of the project, organizational vision and mission statements, goals of the project, and such issues as assumptions, timeframes, and constraints. The next template is a **S**trengths, **W**eaknesses/Limitations, **O**pportunities, and **T**hreats (SWOT), analysis that is designed to help the customer communicate the issues facing them and how this project can solve those issue(s). The third template is designed to help teams build stories which describe the activities the system needs to accomplish. It includes the stakeholders, the system description, and the workflow that occurs. The next part of the template helps translate the story to a database design and a set of screens that will compose the project. The first leads a team through the process of entity relationship modeling and database design. The next two form the basis of sketches for the screens needed and then the detailed layout of those screens. The templates include a Data Dictionary section which contains a table design for each table within the database design. It includes table names, primary keys, foreign keys, and attributes for each table. Finally, the template includes a KanBan Activity section which provides a form to store KanBan entries to insure we have a record of all problems encountered and solved during the project. Not only do these templates insure the project is designed and built in a systematic and professorial manner, it builds needed skill in students for use in future projects in industry or government employment.

7. CONCLUSION

This paper describes the process we use to train and educate students in building enterprise information systems. We focus not only on the software development skills needed to build the system, but on the team building and leadership skills needed to function in industry or government work. We have developed templates and standards that can be used to help guide students and document their efforts in the process. We believe these classes provide valuable skills to the students who complete these courses.

8. ACKNOWLEDGEMENTS

We would like to acknowledge the work of the following individuals on the software included in the JSIM.dll. Matt Kruse, Rushi Naik, John Prablu, Christopher Scott Lusk, and Richard Akers.

ISSN: 2167-1435
v29 n1965

9. REFERENCES

Brewer, J. (2005). Project managers, can we make them or just make them better? *ACM/SIGITE Conference Newark, NJ* pp.167-173.

Churchman, W. (1997). Plenary Address, Association for Information Systems.

Covey, S. R., & Franklin Covey (Firm) (1994). Principle-centered leadership. Franklin Covey, Provo, Utah

Cyrillo, M. (2011). Lean UX: Rethink Development, *Information Week* November 24, 2011, pp. 40-44.

Jone J. L. (2012) "Grace Hopper Quotes." About Women's History. Retrieved 24 June, 2012 from http://womenshistory.about.com/od/quotes/a/grace_hopper.htm

Jonker, C., Popova, V., Sharpanskykh, A., Treur, J., Yolum, P., (2012) Formal framework to support organizational design, *Knowledge-Based Systems*, Volume 31, July 2012, Pages 89-105, Retrieved 28 June, 2012 from (http://www.sciencedirect.com/science/article/pii/S0950705112000512)

Kim, S., Park, S., Yun, J., and Lee, Y. (2008). Automated Continuous Integration of Component-Based Software: An Industrial Experience. P*roceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering* (ASE '08).

Philippe Kruchten. (2011) A plea for lean software process models. *Proceedings of the 2011 International Conference on Software and Systems Process* (ICSSP '11).

Maslow, A. H. (1943). A theory of human motivation. *Psychological Review*, *50*(4), 370-396. doi:10.1037/h0054346

Maxwell, J. C. (2011). The five levels of leadership: proven steps to maximize your potential. Array New York: Center Street.

Raccoon L. B. S. (2006). A leadership primer for software engineers. *SIGSOFT Software Eng. Notes* 31(4), 10-15.

Shneiderman, B., and Plaisant, C. (2010). Designing the User Interface: Strategies for Effective Human-Computer Interaction, 5th Edition. Addison-Wesley, Boston.

Yukl, G. (1994). Leadership in Organizations. Prentice Hall, New Jersey.

US Army (2000), *US Army Field Manual 7-20*, HQ United States Army, Washington, DC.

Zenkert, L., (1990) personal communication.

.

ISSN: 2167-1435
v29 n1965

Appendices

Appendix A - JSIM.dll

HISTORY: Our JavaScript IFrame Movement Dynamic Link Library (JSIM.dll) has been an ongoing project at The University of South Alabama's School of Computing for several years. Matt Kruse developed the Java script to move IFrames. Rushi Naik and John Prablu adaped thiat code and used it to form the .dll. Christopher Scott Lusk improved the .dll and Richard Akers corrected several issues and added needed features. The entire file is available at USASystemsDevelopement.com for download and use.

DESCRIPTION: The purpose of this library is to allow IFRAME objects to be dragged around the screen in the same way that popup windows or drag-able DIV tags are often used. Since IFRAME objects always cover form objects, this makes an ideal solution for a simulated "popup window" on a page with form objects. The JSIM.dll contains the following custom controls:

Drag IFrame
Open IFrame Button
Open IFrame Link Button
Open IFrame Image Button
Open IFrame Grid View
Close IFrame Button
Close IFrame Link Button
Save And Close IFrame Button
Checkbox Grid View
Dynamic Menu
Radio Button Grid View

The following are some of the Data Management Controls for the SV Table contained in the .dll:

CreateSVTable(String) As Boolean
InsertSVTableValue(of T)(String, ByRef T) As Boolean
GetSVTable() As System.Data.DataTable
GetSVTableValue(Of T)(String) As T
GetType() As System.Type

ISSN: 2167-1435
v29 n1965

Appendix B - Human Computer Interface

HCI dictates that we maintain consistency within the project. With several teams working on many different screens that would be merged together, we needed to build consistency in from the start. We developed a Cascading Style Sheet (CSS) file customized to the project and the IFrame.dll. We crafted a CSS reference that described the CSS and showed where and how to use it. This guide defines the different classes in the CSS, and using code samples, shows the members of the teams how to use the CSS within their portion of the project. The use of the CSS and guide helped ensure internal consistency in the project and assumes the class project will be using the JSIM .dll file. Below are examples from the CSS guide. The complete guide is available from our web site USASystemsDevelopment.com

Introduction to CSS
This document serves as a guide line for using CSS in the ISC 567 class project, for the Fall term of 2011 and Spring term of 2012. It will describe the basic high level classes defined in the primary CSS document, and explain when and how to use them. This document will also make note of the areas in the class project that require extra styling that cannot be done automatically via the CSS.
The CSS document being described is called "IFrameStyles.css", and can be found in the Sakai system under project resources. The project management team (a role normally filled by the ISC 568 students) will be responsible for keeping this document up to date. This CSS document assumes the class project will be using the JSIM .dll file.

Major Classes in CSS
The following major classes in our CSS are meant to be applied to the outer most tag in each ASP.NET page, based on the contents that iframe will contain. Every ASP page will have a <body> tag, which contains a <form> tag, which contains a tag. That first tag is where we will use the following classes.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <link rel="Stylesheet" type="text/css" href="Styles/IFrameStyles.css" />
</head>
<body>
    <form id="form1" runat="server">
    <div class="AddEditIFrame" style="width: 400px; height: 215px">
```

There are a few things to notice in this code snippet. First of all, notice that it shows the proper way to link the CSS file into the ASP.NET page. The other thing to notice is that the size of the AddEditIFrame class is not specified in the CSS file, but in the individual page where it is used. This allows us to use the same class in a large number of windows, without having to make custom classes for every possible size combination. So remember, when using the following major classes, be sure to specify the size with a style attribute in the tag.

ListIFrame
The ListIFrame class is used to style iframe windows that will be used to list records from a database using the OpenIFrameGridView object supplied in the JSIM .dll.

In addition to the CSS class used to define the iframe container, use of the OpenIFrameGridView object requires some extra attributes to be set in order to make it conform to the look and feel of the rest of the site. The following attributes must be added to the <ccJSIM:OpenIFrameGridView> tag:

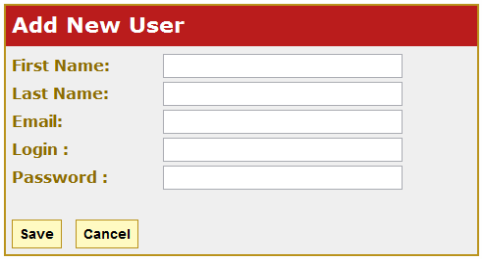
- ForeColor="#333333"
- HighlighedRowColor="#FFFFAD"
- HeaderStyle-CssClass="gridViewHeader"
- HeaderStyle-ForeColor="White"

Additionally, add the following sub tags inside the <ccJSIM:OpenIFrameGridView> tag:

- <AlternatingRowStyle BackColor="#FFFFFF" />
- <RowStyle BackColor="#EEEEEE" />
- <SelectedRowStyle BackColor="#D1DDF1" Font-Bold="True" ForeColor="#333333" />

ISSN: 2167-1435
v29 n1965

AddEditIFrame
The AddEditIFrame class is used to style the iframe windows that allow the developers to add new records or edit existing ones. It doesn't rely on any particular control in the JSIM .dll, it just applies generic look and feel styles to an iframe to make it match the rest of the site.



CheckBoxIFrame and All Other Gridviews
The training modules defined a CheckBoxIFrame class and used it to style the iframe that displays records with the CheckboxGridView class. However, this class is identical to the ListIFrame class, and is completely redundant. As such, we will not be using this class at all, we will use the ListIFrame class to style all windows that use any of the custom grid views associated with the JSIM dll.

The CheckBoxGridView class also requires some additional formatting in order to fit in with the rest of the CSS. The following code must be added to the <ccJSIM:CheckboxGridView> tag for it to fit in with the rest of the color scheme.

- ForeColor="#333333"
- HeaderStyle-CssClass="gridViewHeader"
- HeaderStyle-ForeColor="White"

Additionally, add the following sub tags inside the <ccJSIM:CheckboxGridView> tag:

- <AlternatingRowStyle BackColor="#FFFFFF" />
- <RowStyle BackColor="#EEEEEE" />
- <SelectedRowStyle BackColor="#D1DDF1" Font-Bold="True" ForeColor="#333333" />

These changes are identical to the changes you made to the OpenIFrameGridView object, except that we do not need to specify a highlight row color here.

Other components of the system that use the CSS include:

MainMenuTab
ListIFrame
PanelStyle
AddEditTable
TextBoxLinkButton
MenuOption
PaddedCell
Button

ISSN: 2167-1435
v29 n1965

Appendix C – Training

We have developed a series of training videos to help students learn the basics of using the IFrame.dll, LINQ, and CSS.

A. IFrame DLL Training

The videos that cover the IFrame.DLL are several hours long and also serve as a refresher on N-tier architecture and .net programming.
Part one includes an introduction to the n-tier architecture and setting up a login page, building a database in SQL to form the data tier for the login, and building the data access and presentation tier for the login page. It includes the coding needed to authenticate a user in the project.
Part two builds the data tier and data access tier to manage users in the system. It introduces the concept of using an IFrame and shows how it is used in the presentation tier to build a manage users page. It covers how to use panels and tables within an IFrame, and how to populate a gridview and bind that information from the data tier to the presentation tier. It also covers basic create and delete operations on system users.
Part three covers the data tier, data access tier, and presentation tier for creating new and editing existing users. It includes building the stored procedures in SQL and building the screens in Visual Studio on the presentation layer.
Finally, part four covers building the database, data access, and presentation tier portion required to assign roles to users. This could be used as part of a role based security system. This set of training videos are used to give students the basic skills using iFrames and other tools needed to be productive in working in the project selected by the instructor. The videos are designed to be self paced, but each student completes the system shown by the videos and that work is presented to their team leader and project manager on a weekly basis.

B. Cascading Style Sheets

The Cascading Style Sheets training video covers the advantages of using CSS and how to use the CSS file developed for the project. This video also provides training on the use of master pages and child pages.

C. LINQ

The Language Integrated Queries (LINQ) training video introduces LINQ, data sources, data types, data context, and the pros and cons of using LINQ. It also examines using query syntax vs Lambda Expressions.
All these training materials are available on our web site at USASystemsDevelopment.com

Appendix D - Digital Rules of Engagement

Introduction
Any large software development effort will present integration issues that if not resolved can doom the project before it even begins. This Digital ROE is an effort to identify and resolve these issues before they impact the development cycle. Since new issues will arise as we go forward, this is a living document that we can pass on to the next group in hopes that they can benefit from our experiences. We should strive to make this project look like it was crafted by the same person. It should be seamless and effective.

Naming Conventions
Use of Standard names for software objects is important. Use the form we have used in previous classes, for instance, a label for a text box for entering a person's first name would be call lblFirstName.

Languages
All coding should be VB for the Presentation tier, C# for the Data Access and Business Logic Tiers.

Connection String
A standard connection string and database user and password is critical in ensuring your calls for data access methods and functions work correctly. The user is "castuser" and the password is "castpass". It is imperative that you use this user. Eventually, the database will be locked down so that only this user may access the database, and this user will only be able to run stored procedures. This will be the connection string that all team will be required to use. Feel free to copy and paste this into your projects.
The connection String name is CASTConnectionString

```
<connectionStrings>
      <add name="CastData.Properties.Settings.CASTConnectionString"
         connectionString="Data Source=CID;Initial Catalog=CAST;Persist Security Info=True;User
ID=castuser;Password=castpass"
         providerName="System.Data.SqlClient" />
   </connectionStrings>
```

Cascading Style Sheets
CSSs have been developed for aspects of the project. To insure we are internally consistent into the project we must all use the same CSS and same version. The current CSS file is loaded on the Class website under resources.

Database Use and Updating
Always, ALWAYS use the most current version of the database script. Your code must work with the database script that is on the project site. Any errors in the script, or changes and additions you would like to suggest can be emailed to Curt Parker (cwp901@jaguar1.usouthal.edu).
When using the database, special care should be taken to minimize direct access to the database tables. Nearly all of your database access should be done using stored procedures. If a stored procedure is not sufficient, feel free to update it or create a new one for your use. However, in order for your procedures to be permanent, you have to tell Curt Parker what you did so it can be added in the global database file. The stored procedures must be in the CAST.sql file or they will be erased the next time you update your database.

Software Reuse
Endeavor to code in blocks that can be reused in another part of the project.

Screen Size
Screens should be coded to a max size of 950 X 650 pixels

ISSN: 2167-1435
v29 n1965

Font Type and Size
See CSS and HCI Guideline documents in the Resources folders.

Comments
In any project where you have multiple teams working on the same pieces of code, comments are crucial. They allow different developers to come behind you and take a glance at the code that you have written and get an understanding of its intent fairly quickly. David Njoku has a very good article on why comments are important here: http://allthingsoracle.com/how-to-make-comments-the-most-important-code-you-write/ (even though it is about Oracle, the substance of the article can be applied to our current project)
While working in the project, if you come across a piece of the code that you have written or that you are familiar with, and it is not commented properly, take the time to leave comments. This will assist future teams who are trying to understand the project.

Formatting Code
Along the same lines of comments being important, correctly formatted code is almost as important. Formatted code is not only easier to read, it is also easier to debug whenever you have an issue. Make sure that when writing your code, you not only use correct spacing and indentation, you also use the correct capitalization of your variables and data objects (Good: lblFirstName Bad: lblfirstname Worst: Label1). Also do not forget, there is no punishment for having extra spaces in your code. Group together items that make sense to be grouped together (such as variable declarations, functions and procedures that are used together to perform some goal, etc.) and separate them with space. This will go a long way in making your code more readable and easier to understand. Using correct formatting now will cause fewer headaches down the road whenever your projects become large and complex. Also, whenever you enter the professional working world, incorrectly formatted code makes you look like an amateur, and will cause your colleagues to lose faith in your programming abilities.

ISSN: 2167-1435
v29 n1965

Appendix E – Kanban Sheets are used to collect data that is then input into the KanBan system.

Data is is entered using one of the pull down menus, or in the problem and description sections, entering text.

KanBan Activity Record	
Team	Choose an item.
Initials	Choose an item.
Lifecycle Step	Choose an item.
Type of Activity	Choose an item.
Priority	Choose an item.
Status	Choose an item.
Problem:	
Description:	

Status

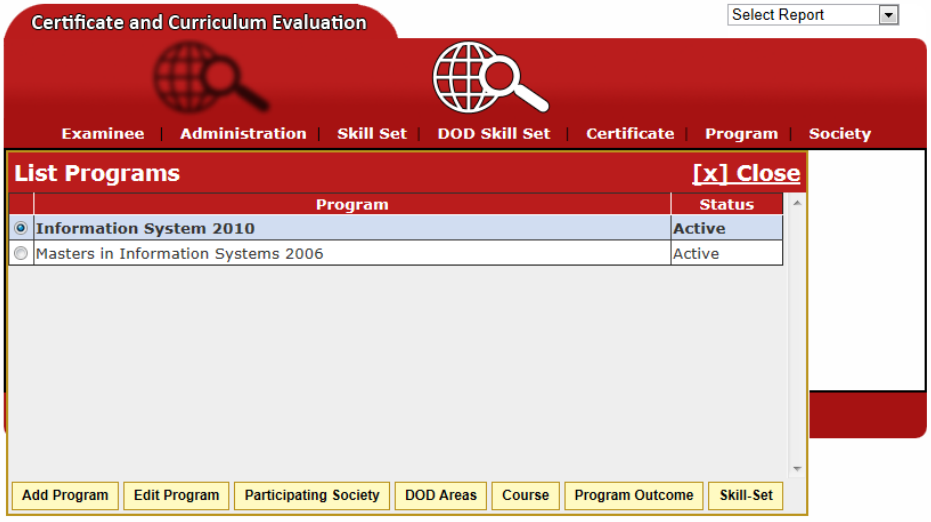
Choose an item.
Staging
In Progress
Complete
Build
Deploy

ISSN: 2167-1435
v29 n1965

Appendix F – Screen Shots of Developed System:

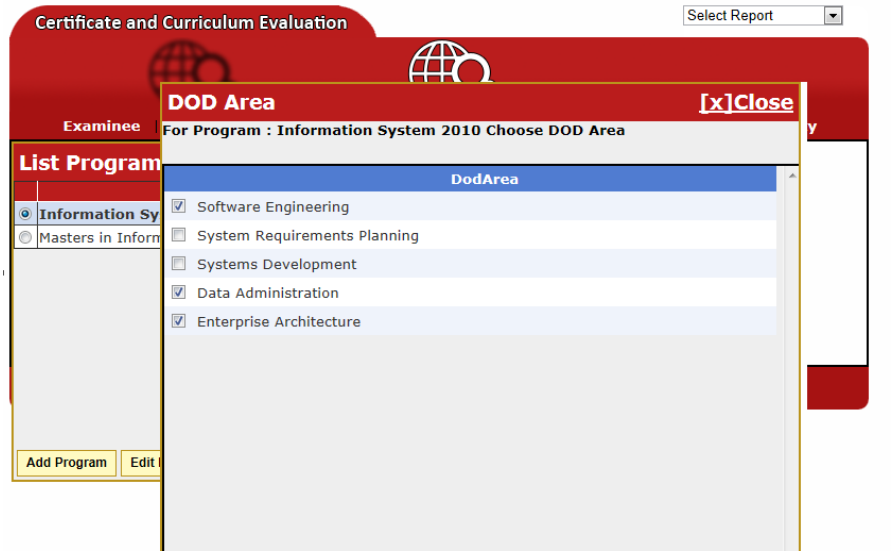
This is the main menu for our developed system. All functions within the system can be accessed directly from this menu. Any reports general to the whole system can be accessed from the "Select Report" Drop-down.

The program function from the main menu was activated by pressing the "Program" button. Notice that the page "List Programs" was positioned on top of the "Menu. Both the "Menu" and the "List Programs" pages are displayed in IFrames. Note also consistency of colors. This was accomplished by the CSS. The buttons at the bottom of the "List Programs" page are used to access subsequent pages as needed.



Please notice the button "DOD Areas". The intent for this button is to bring up a page for a selected program. The "Information Systems 2010" program was selected by activating the radio button—the line turned blue in response. Once the program was radio button selected, the following page "DOD Area" can be activated by pressing the "DOD Area" button—This selected page will show the DOD Areas relevant to the Information Systems 2010 curriculum. Even though the "DOD Area" page is displayed, the "DOD Area" page was carefully positioned to enable at least part of the "List Programs" page to be viewed. This anchoring effect makes it possible to remember the navigation used to descend further into the application.

ISSN: 2167-1435
v29 n1965



The "DOD Area" screen executes a "check-box-select" function. Elements of a list may be assigned to an element of another table. In this case, DOD Areas may be assigned to the Program Information Systems 2010. According to this graphic, Software Engineering, Data Administration, and Enterprise Architecture have been assigned to Information Systems 2010.

This screen shows a three level from Program to Course to Course Outcome. A specific course outcome is being edited to depressing the "Edit Course Outcome" button. After editing is performed, the "Edit Course Outcome" screen will be closed and the three screens List Program>List Course>List Course Outcome will remain.