Cintas Skills Experience and Tools Application (SkillET)

By

Richard Hill

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Computer Science Technology

University of Cincinnati
College of Applied Science

May, 2005

Cintas Skills Experience and Tools Application (SkillET)

By

Richard Hill

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment of the Requirements
for
the Degree of Bachelor of Science
in Computer Science Technology

Richard Hill	Date
Prof. Tom Wulf, Faculty Advisor	Date
Pat Kompf, Department Head	Date

Acknowledgement

I would like to extend a special acknowledgement to the Cintas Corporation for their aid in funding this project and providing the resources to use during its execution. I would like to specially recognize members of their Application Development Consulting Team. These members are Linc Kroeger, Matt Arnett, Casey James and Lahai Karmo. They were all invaluable resources during the development of this project and have truly made working at Cintas a unique and rewarding experience. Their input for design and recommended features was critical in producing a quality product. I would also like to acknowledge Christopher Judd, whose mentoring has made me a far better developer.

Dedication

To Maria – without your constant support and encouragement I wouldn’t be here today.

Table of Contents

List of Figures

Abstract

In the constantly changing business environment of today it is becoming increasingly necessary for businesses to be able to respond quickly to change. In order to adapt to these changes a business must be able to determine how adaptable it is at any given time and where its resources are. This paper discusses an application designed to monitor and display how adaptable teams are in relation to tasks that they need to perform. This paper discusses the development of the SkillET application used to capture data on the flexibility of teams in the MIS department of Cintas Corporation and the functions of that application. I discuss the benefits realized by the business in using this application as well as the underlying technology that is used to build it. These technologies include a Web interface for the application and an extensible scalable architecture that allows the SkillET application to grow over time.

1. Statement of the Problem

The only thing constant is change. It is true in life and also for today's information technology projects. In order to be able to succeed a company must be able to change their strategy at any time to meet the changing needs of their customers and the current business climate. The second law of Lean Six Sigma is *"The velocity of any process is proportional to the flexibility of the process*[5]*."* Realizing a need to improve their flexibility and versatility, Cintas requested a system to aid in measuring and analyzing the relevant pieces of data for this metric.

Understanding the flexibility and versatility of team members in an Information Technology (IT) group is necessary to understanding how they contribute to given problem domains and the changes that naturally occur within those domains. The Six Sigma process for quality control has been shown to improve business functions and has become a standard in many modern businesses for process improvement[5].

The Six Sigma method for quality control lists a series of five steps to improving business functions: define, measure, analyze, improve and control. We have already defined the area of improvement as the flexibility of our IT groups. The next step is to measure. In order to accurately measure flexibility we must have a system in place to track the key indicators that make up flexibility. These key indicators can be found in the forms of employee skill sets, previous experience in a given problem domain and familiarity with tools used in solving relevant problems.

Cintas has recognized that the members of IT groups will benefit from the Six Sigma process. Team members will only be given tasks to complete for which they are trained. Expectations of quality and output are more subjective based on the presence or

absence of skills. Team members will also have more professional and personal growth opportunities because a clear training path will be present. Customers will benefit from the system thanks to decreased wait times for project completion and higher quality output. The company benefits by realizing the increased profits of leveraging each employee's particular skills.

2. Description of the Solution

I propose to create the SkillET application for Cintas to allow the company to measure the individual components of flexibility as a means to improving overall business process. Cintas will also be able to realize improvements in tool dependency. This application would allow managers to link the tools used directly to an application. The documentation of this development path will provide additional value that goes beyond the scope of the system.

This will be a Java Enterprise Edition (J2EE) Web based application that utilizes existing standards in Cintas' development process. The application will use a combination of various Java enterprise technologies to present an easy to use interface for standard users and managers. SkillET will also provide a means to persist the users' information for later retrieval and reporting functions to view said information.

It is also worth noting that this system would not necessarily have to be strictly limited to Information Systems. The potential exists to use the system in other areas of the company and other lines of business to improve efficiency through flexibility. For example, tracking which employees with forklift certifications, high bay experience and embroidery experience could be useful for managers in distribution centers and manufacturing plants. Skills could also easily be tracked for accounting, finance, logistics, et al.

2.1 User Profiles

Two types of users will exist within the SkillET application, standard users and managers.

2.1.1 Standard User

This type of user will be able to view a list of all of the members of the teams he/she belong to. This list will not reveal any information about team members except for their names. Users will be able to view a list of all of the tasks that are assigned to their team. The list of tasks will also show the users flexibility score for that particular task. A list of prerequisites that belong to team tasks will be visible to the user. From this screen the user will be able to identify which tasks a given prerequisite is associated with as well as indicate whether or not the user possess that prerequisite. Finally, a user will be able to view a list of all of the skills that belong to the team's tasks. Much like the prerequisite screen the user will be able to see which tasks a skill is associated with and update whether or not the user has that skill. Additionally the user will be able to view the flexibility percentage that each task has toward its associated task.

2.1.2 Manager

This type of user will be able to view a list of the members of their team as well as define the team members and the team name. Managers will be able to create new tasks for their team as well as the skills and prerequisites that are associated with the tasks. This level of user will also be able to define the percentage flexibility that the skills have towards their associated tasks. Managers may also view a list of all of the team's tasks in a visual format through an additional Web page and will be able to generate a printable report that contains the information included on that page.

2.2 Design Protocols

2.2.1 Organizational Scheme

The organizational scheme for this application mimics the organizational scheme of the underlying data objects that are used in the application. The application is split up into different and distinct sections that are specifically geared toward the objects that are being manipulated. Separate screens are used for the viewing and maintenance of teams, tasks, skills and prerequisites. In each section normal users will be able to view a list of all of the instances that are available to them and update their profile based on those instances. Team leaders will be able to create, edit and delete these instances as they go through the same screens.

2.2.2 Interface Design / Navigation

User interface and design is simple and consistent throughout the application. In sticking with the standards that are already in place for Cintas, users are able to adapt to the operating environment more quickly and take full advantage of the potential of the application.

There is a menu on the left side of the screen with four different options, one for each of the types of instances that are used in the application. Entries are for Teams, Tasks, Skills and Prerequisites (figure 2.1).

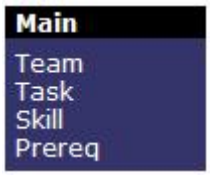


Figure 1 - Left Menu Bar

By clicking on any of these menu options, the user is taken to a list of their current status regarding these objects. Users are able to update their completion of skills and prerequisites by using a check box on the listing pages of those two data types.

Team leaders have the same basic functionality with a few enhancements. Leaders are able to select any of the objects that apply to the teams they lead. This leads them to the interface for editing that existing instance. Leaders also have an "add object" link on the top and bottom of each listing. They may use that link to add new instances of that type to the system and attach them to appropriate existing instances. Leaders are also able to delete an existing instance by selecting the delete icon from the listing page or by using a delete button on the update page. The largest difference for team leaders is that they can observe the completion of all users on their teams as well as themselves.

2.2.3 Icons / Graphical Symbols

The SkillET application will use several common icons throughout the application to allow users to navigate and perform actions more quickly. These icons will be placed consistently directly to the right of the object that can be edited or deleted and surrounding lists of objects that can be added to.

 A pencil icon indicates that a user can edit an object

 A plus sign indicates that a user can add a new instance of an object

 A trash can indicates that a user can delete an object

2.2.4 Color Scheme

The color scheme used in this application will be based on a standard cascading style sheet that has been listed as the Cintas standard for colors and formatting. The color scheme generally involves the use of navy blue, white and grey. These colors are used

because the low intensity of the colors allows the user to be in the application for long periods of time without experiencing eye strain.

2.2.5 Help

Users of this application will be able to access a help menu from a “help” link at the top right corner of each page. The placement of the link inside of the image at the top will remain constant throughout the application and will have different content depending upon the screen that it is accessed from.

3. Objectives of the Project (Deliverables)

Certain items have been defined as necessary for the successful completion of the SkillET application. The components present in the final version are listed below.

1. A Web based application that allows users to view the teams that they belong to as well as their ability to support tasks that are assigned to that team.
2. Team leaders are able to view the members of their team as well as add and delete members to and from a team.
3. Team leaders are able to define team tasks and the skills and prerequisites that compose them.
4. Standard Cintas Web application security, which includes authentication against Active Directory and application level authorization based on Active Directory Security Groups.
5. Team leaders have a visual representation of all of their team members' ability to support team tasks by static images that approximate user task coverage.
6. Team leaders are able to click an icon to generate an Excel spreadsheet that contains the currently captured data for any/all of the teams that they lead and all of the data pertaining to those teams' users.
7. The data for this application is stored in a SQL Server 2000 database on the backend.
    a. Data from this database is accessed using Hibernate[3]
        i. Hibernate is an open source Object Relational Mapping Framework that uses xml files to translate between Object Oriented business objects and a relational database.

ii. Hibernate xml files are generated at compile time using XDoclet. XDoclet locates specific comments in the application and use this as a basis for generating xml files.

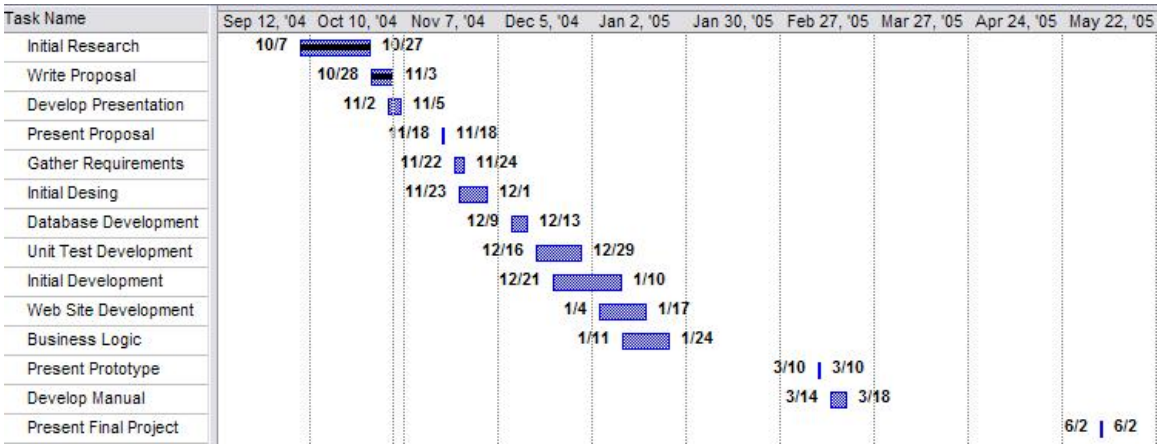
8. The application design is flexible, scalable and verifiable.
    a. Test Driven Development (TDD) ensures that the application is verifiably stable before each release
    b. The application design premises are based on a combination of design patterns including Factory, Abstract Factory, Data Access Object, Data Transfer Object and Model-View-Controller (a.k.a. Model 2).
    c. The Model-View-Controller pattern is handled by the Struts Web application framework
    d. The view layer is easily modifiable and reusable by using the Struts Tiles plug-in.
    e. The data submitted by the user is verified using the Struts Validation framework
    f. Data from the application is logged using the Log4J framework and the Jakarta Commons Logging framework.

4. Design and Development

4.1 Budget

	Name	Quantity	Explanation	Cost
Hardware	Production Server	4	Need	$3700.00
	Preproduction Server	3	Need	3000.00
	Development Server	1	Need	1800.00
Software	BEA WebLogic	4	Licensing	17000.00
	MS SQL Server	1	Licensing	19999.00
	Borland JBuilder	1	Licensing	3500.00
	Borland OptimizeIt	1	Licensing	2600.00
	Apache Ant	1	Need	0.00
	Jalopy	1	Need	0.00
	Hibernate	1	Need	0.00
	XDoclet	1	Need	0.00
	Jakarta Struts	1	Need	0.00
	Jakarta POI	1	Need	0.00
	Jakarta FOP	1	Need	0.00
	Log4J	1	Need	0.00
	JUnit	1	Need	0.00
Total				122699.00
Already Placed				120899.00
Actual				$1800.00

4.2 Timeline



4.3 Software

4.3.1 Java

I will be using the Java programming language as the core of the system. Java's Object Oriented design and best-of-breed approach allow for a robust solution that is extensible and easily modifiable. This will decrease the total cost of ownership (TCO) by improving maintainability and scalability. The Java programming language has many built in components that can be utilized to speed up development time. Enterprise Java Beans will be used to help architect a Service Oriented Architecture (SOA). JavaBeans will enable Web pages to use objects as an abstraction. Java Server Pages (JSP) is a technology to use server side rendering for dynamic Web content. Custom tag libraries can be developed to aid in the development using JSP. The Java Standard Tag Library (JSTL) provides a myriad of methods that allow for HTML like syntax with the power of Java behind it.

4.3.2 BEA WebLogic Server

The BEA WebLogic Server is a proven leader in the realm of Web application servers. Its high performance and reliability allow it to stand out amongst its competitors. The built in clustering support for this server also allows for easy setup of complex network configurations. Integrated connection pooling also allows this server to give faster database access to the system. The current version of WebLogic Server uses the JRockIt Java Virtual Machine (JVM) which is also developed by BEA. This virtual machine supports Java 2 Standard Edition and Enterprise Edition version 1.4.2.

4.3.3 Microsoft SQL Server

The performance and scalability of SQL Server has made it the choice for this system. SQL Server comes in at a lower price than many of its competitors and its simple administrative interfaces allow for lowered maintenance costs from an operational standpoint. It is possible that the system will move to an Oracle database in the near future, so any development features that are specific to SQL Server will be thoroughly documented.

4.3.4 Borland JBuilder

Borland's JBuilder suite will be use as the primary Integrated Development Environment (IDE). It has built in UML design that allows developers to "draw" out their code. In addition, Borland's OptimizeIt lets developers track the performance of an application to fine tune performance and more easily track programming errors.

4.3.5 Apache Ant

Apache's Ant is the leading script based build tool for Java. Ant will let developers build and deploy the project consistently and simply. Its task based build scripts allow development time to be more efficiently utilized.

4.3.6 Jalopy

Jalopy will be used as part of the regular Ant build cycle to ensure consistent formatting throughout the source code used in the project. Jalopy allows strict rules to be placed on how source code is formatted and can even automatically add documentation where specified.

4.3.7 Hibernate

The Hibernate Object Relational Mapping (ORM) framework allows developers to bypass the creation of complex data access objects and data transfer objects by automatically converting persistent data from the database into user defined instances. The improvement in development time from this framework is substantial. Hibernate will be using XML files that are generated at compile time to determine how the mapping of objects to the database models will be done.

4.3.8 XDoclet

XDoclet uses the JavaDoc specification to dynamically render XML files based on comments written with Java source code[6]. In the case of the SkillET project, XDoclet will be generating the XML files that Hibernate uses to map objects to tables in the database.

4.3.9 Jakarta Struts

The Jakarta Struts framework allows developers to more easily architect a Web application that enforces the Model-View-Controller Pattern. This will allow the system to be built quickly with easy maintenance. Struts also had the added benefit of built in support for input validation and a complex page composition system.

4.3.10 Jakarta POI

The Jakarta POI API will be leveraged to dynamically generate Microsoft Excel spreadsheets to be used for reporting.

4.3.11 Jakarta FOP

The Jakarta FOP API will be used to dynamically generate Portable Document Format (PDF) files used in reporting and Scalable Vector Graphics (SVG) to render real time images used for quick overviews of statistical data.

4.3.12 Log4J

Log4J will be used as the primary logging framework throughout the system. Log4J allows developers to easily mark logging issues at different levels and direct the output to a number of different places based on system configurations.

4.3.13 JUnit

JUnit is a unit testing framework that can be leveraged as part of the development process. JUnit tests will be built to ensure the quality of the code that is written and to make sure that no tangential pieces of the project are broken by changes to other areas. Cactus[2] by Apache will be used to imitate a running server during the testing phase and the StrutsTest project can help ensure that user interface requirements are held.

4.4 Hardware

The system will be hosted on an existing cluster of Itanium servers. The servers are currently running dual processors at 3.2 GHz with 8 GB of RAM each. There are currently 3 servers in the cluster with an additional control server. Currently, hosted systems are using less than 20% of total capacity. Preproduction environments are also readily available to mimic production. Development needs include a scaled down version of the same environment. There will be a need for an additional development server to mimic the production environment. A single processor 2.8 GHz Itanium Server

with 2 GB of memory will be sufficient. Production and development database servers are already in place to handle storage needs.

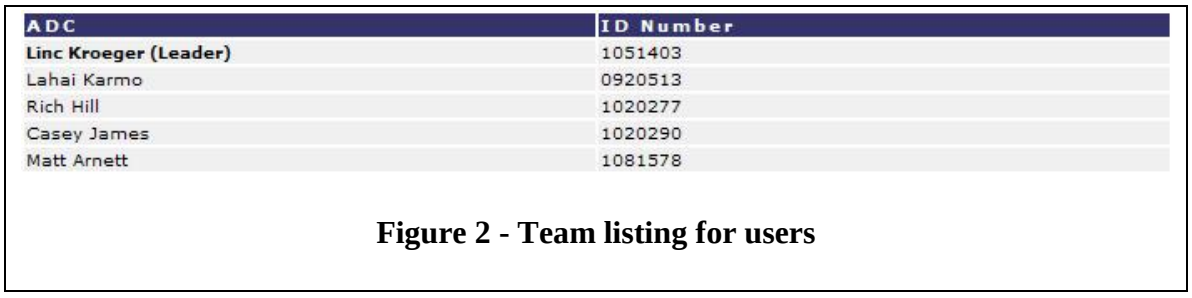
5. Proof of Concept

This section details how the deliverables of the project were met and any problems that were encountered.

5.1 Basic Functionality

SkillET has been built as a Web application from the beginning. It is packaged as a deployable .ear file during its java compile stage with deployment descriptors and Enterprise Java Beans (EJBs) packaged with it. Further it utilizes the Struts Web application framework to make the building and maintenance of the application easier. It has been successfully deployed to a BEA WebLogic server cluster and could be easily modified to work on any Web application cluster that supports EJBs and Java Naming and Directory Interface (JNDI) lookups.

Team members are able to view all of the members of the teams that they belong to upon first entering the application. The default entry screen is a list of the user's teams and all of those teams' members (figure 2).



ADC	ID Number
Linc Kroeger (Leader)	1051403
Lahai Karmo	0920513
Rich Hill	1020277
Casey James	1020290
Matt Arnett	1081578

Figure 2 - Team listing for users

A user of the application may also view their ability to support the tasks of their team by clicking on the "Task" link on the left menu bar. This will direct them to a page that lists all of the tasks that belong to the teams that they are on as well as their

computed flexibility for each of those tasks. This screen does not show the flexibility of any of the other members of the team, only the currently logged in user (Figure 3).

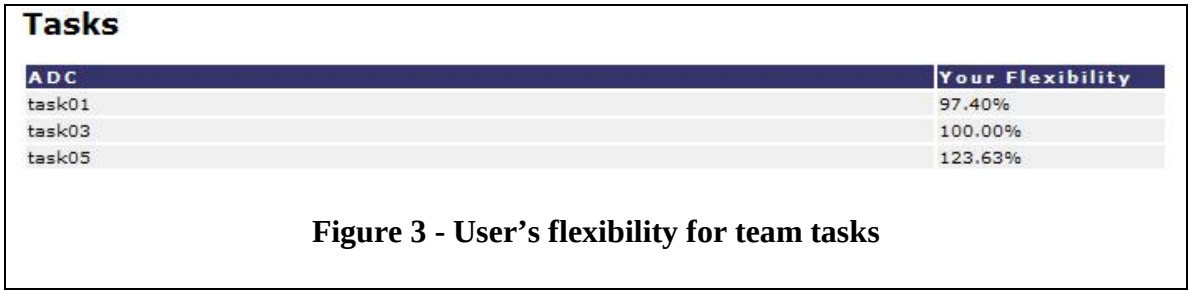


Figure 3 - User's flexibility for team tasks

Both the lists of teams and the lists of tasks are displayed on the screen by placing a collection of the objects into the user request and then iterating through those lists using JSTL.

5.2 Team Editing by Team Leaders

The members of each team are managed by a team leader. In order for team leaders to manage their team there must be an easy way for the team leaders to add new team members. Team leaders are shown the same lists of teams and team members when they enter SkillET, except team leaders are able to click on a link to edit that team or to create a new team (Figure 4).

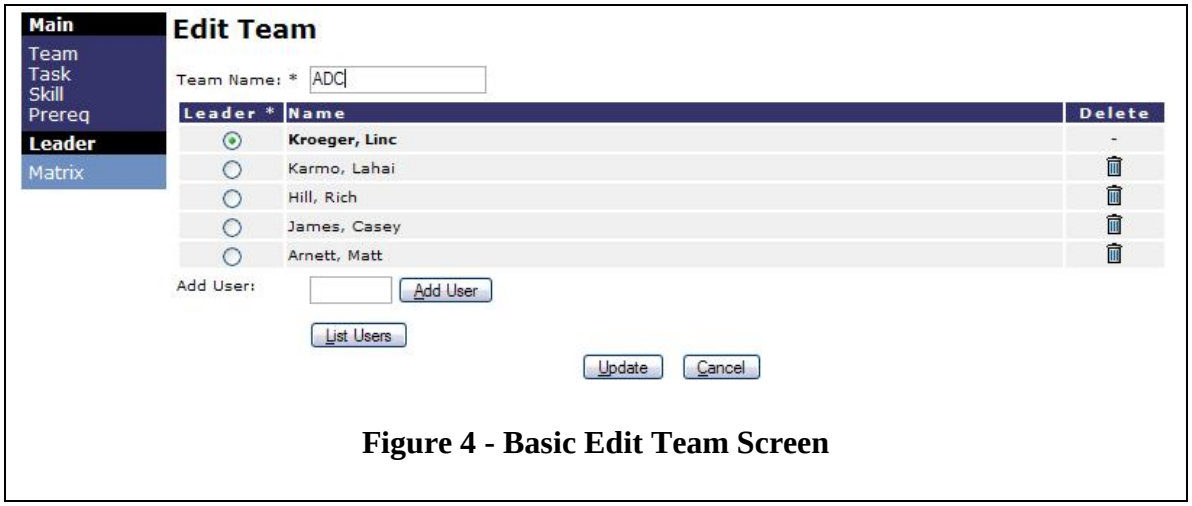


Figure 4 - Basic Edit Team Screen

Once the leader has selected to edit a team they are able to delete team members by selecting the delete icon for the user. That user will still appear in the list of users. However, the user will be moved to the bottom of the list, their name will appear with a strikeout line through it and their row in the table will switch to having a grey background (Figure 5).

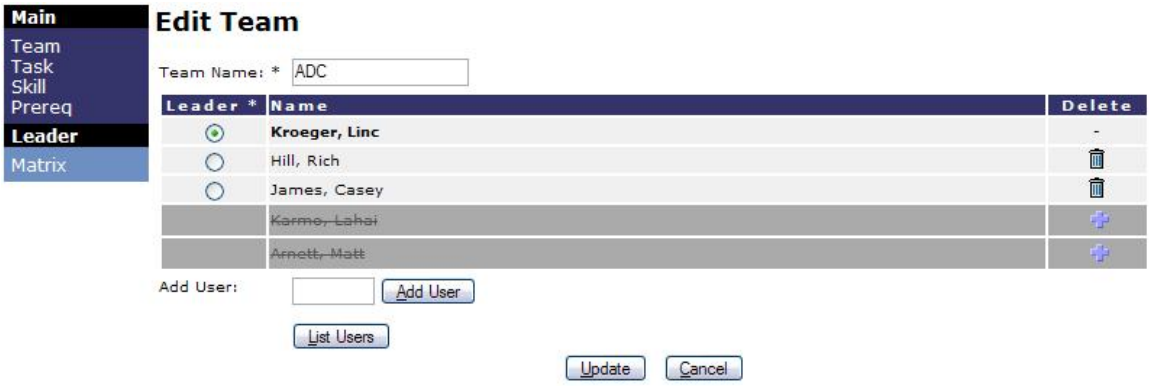


Figure 5 - Team Edit Screen with Two Users Removed

The same user may be re-added to the team by clicking on a plus sign that will replace the delete icon for that user. New team member may be added by typing their user ID into a text box and then clicking the "add user" button. Since this method requires knowing a user's ID number, which is a 7 digit number and may be hard to remember, the leader may also press a "list users" button (Figure 6).

Figure 6 - Add User Text Box Replaced by Drop Down List

This button will replace the text box for user IDs with a drop down list that includes all of the users at the team leader's location. The location is determined by the team leaders active directory attributes and the users are returned by querying Cintas' Active Directory. Because the edit team screen is complex to render and it requires a great deal of back end processing, I decided to lower the burden on the system by using Asynchronous JavaScript and XML (AJAX). Using AJAX I was able to make a call to the Web application server to request the list of users to put in the drop down box and replace the drop down box without having to refresh the page and without having to send that list of users in the initial request object in which the page was loaded. The changes to the team are saved when the leader clicks the update button.

5.3 Task, Skill and Prerequisite Operations

Tasks are the most important part of the SkillET application and team leaders must be able to manage them. As stated previously, tasks are comprised of skills and prerequisites. A team leader must be able to create, view, update and delete all of these types of objects from their team. A team leader is able to create tasks for their team by clicking on the task link in the left menu bar. A team leader is then able to view the tasks for teams that they belong to, just like team members, and also the tasks for all of the teams they lead (Figure 7).

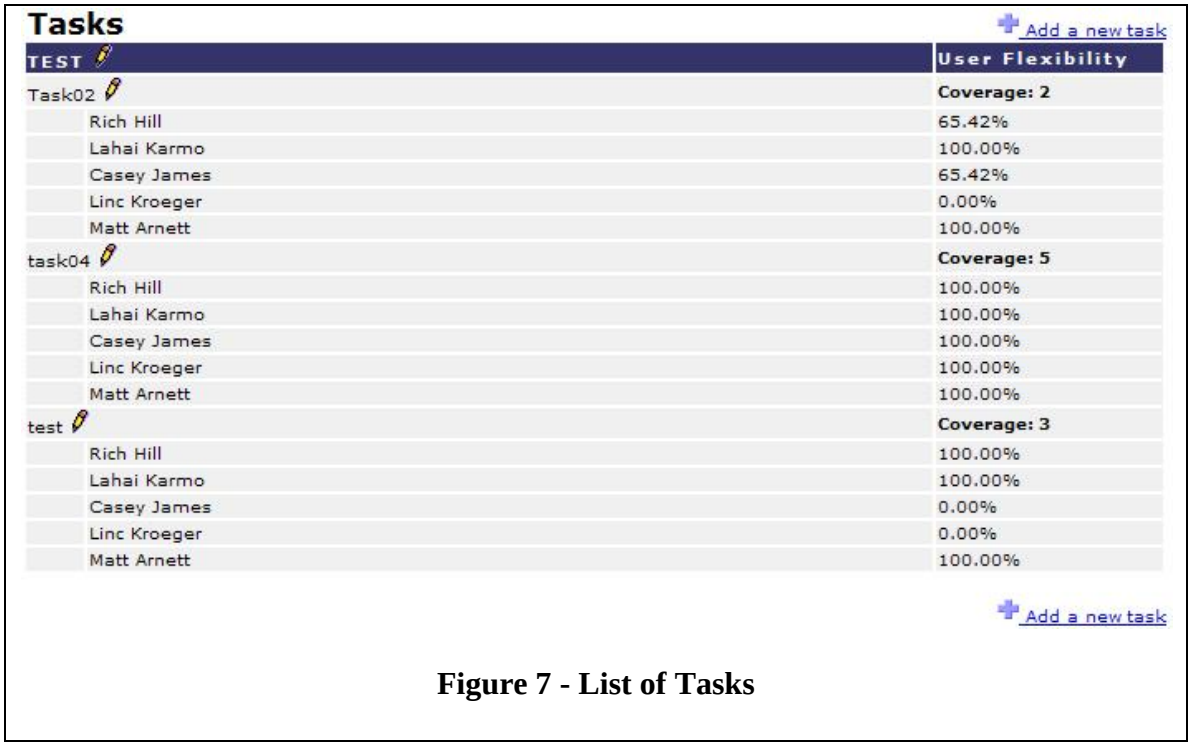


Figure 7 - List of Tasks

Team leaders are able to view the flexibility of each of their team members for all of the team's tasks from this page in addition to viewing the coverage for each task. From this point a team leader may click on a link to create a new task or to update a task that belongs to a team that they lead. Once updating a new or existing task the team

leader is able to add prerequisites to the task that are associated with their team (Figure 8).

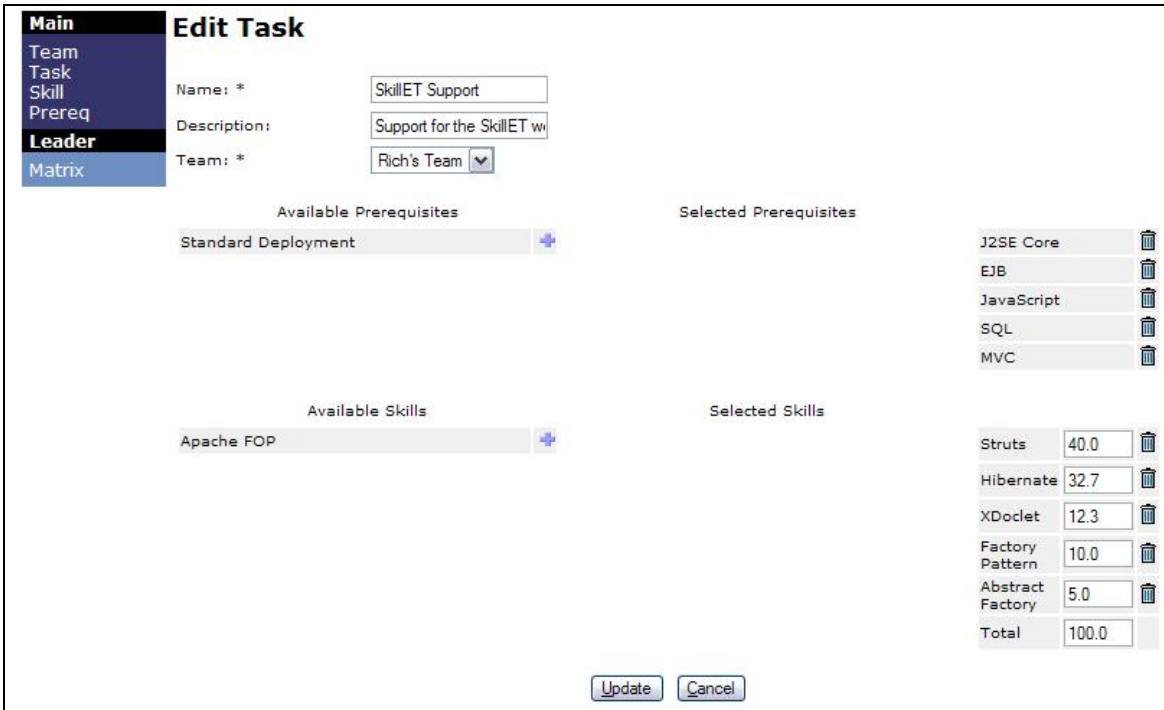


Figure 8 - Basic Edit Task Screen

They can also change the weight of any skills that have already been associated to that task. Changes to the task are saved when the leader clicks the update button.

Leaders are able to create prerequisites through roughly the same mechanism after clicking the prerequisite link on the left menu bar. A list of prerequisites is presented to the team leader and they are able to delete prerequisites from this screen (Figure 9).

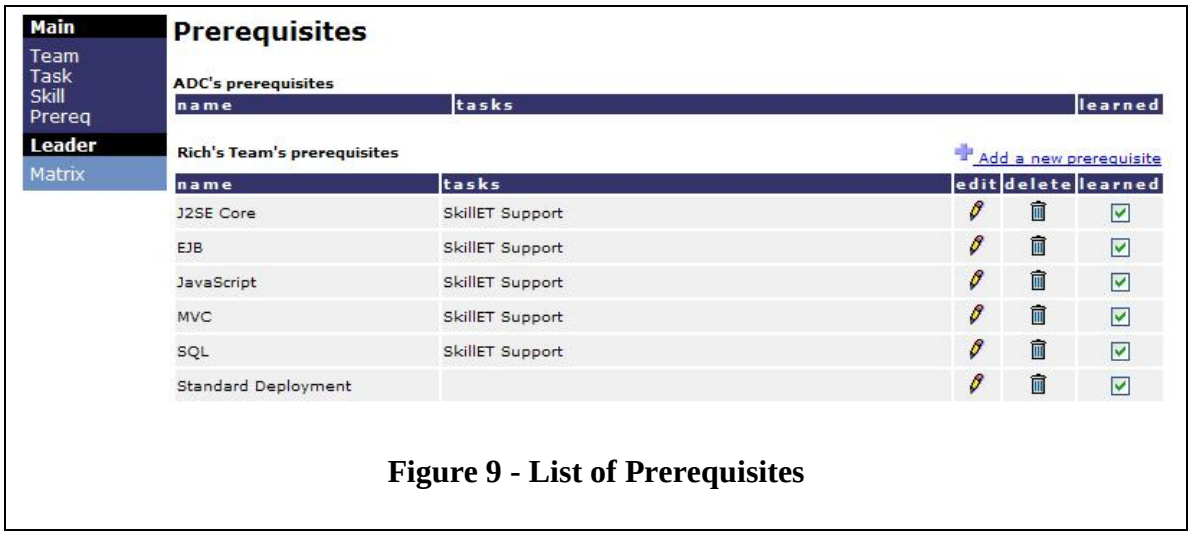


Figure 9 - List of Prerequisites

From this screen team leaders are also able to indicate whether or not they possess a prerequisite by using the checkboxes that correspond to the prerequisite on the same line. Adding and updating prerequisites takes place through different links, although it presents the team leader with the same form for each operation (Figure 10).

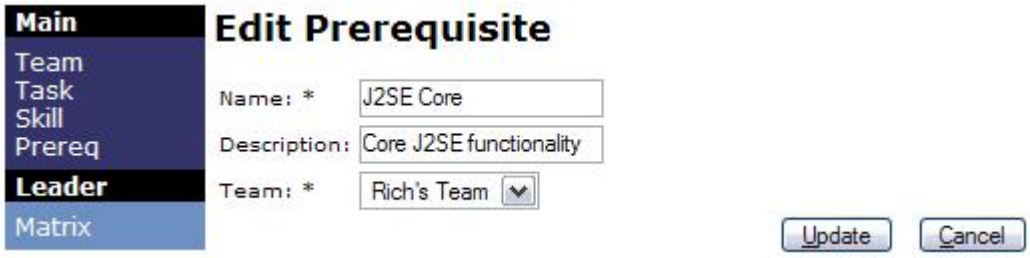


Figure 10 – Basic Edit Prerequisite Screen

Prerequisites, however, are much simpler objects and as such have far fewer parameters that must be defined by the leader when creating or updating the.

Skills may also be created and updated in the same way after clicking on the skills link in the left menu bar. Again, a list of skills for the teams that a team leader is on or manages is shown (Figure 11).

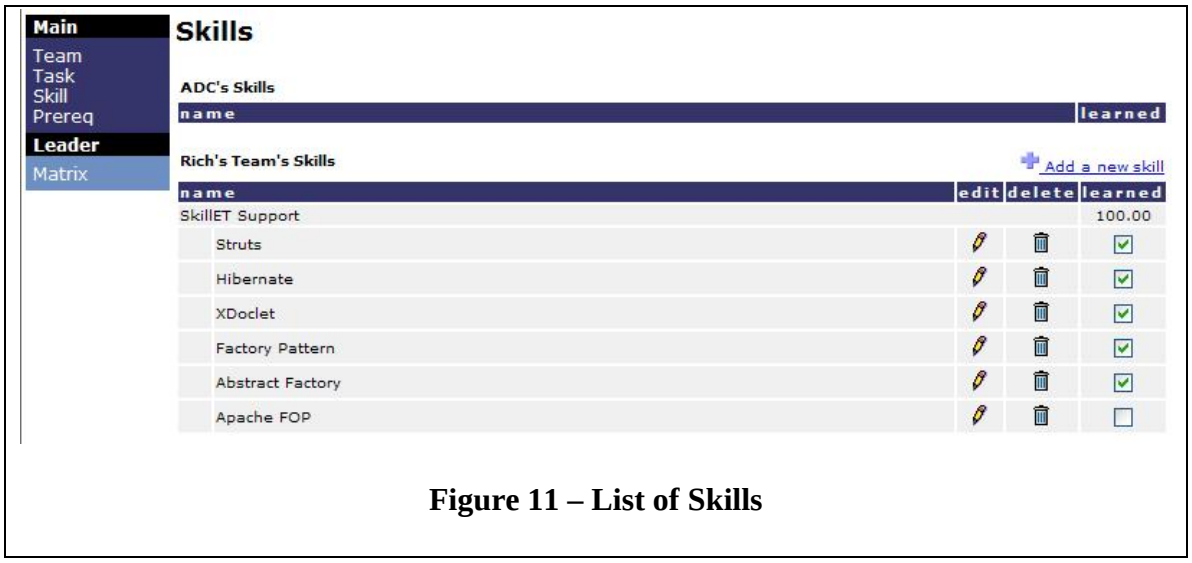


Figure 11 – List of Skills

From this screen a team leader can update whether or not they possess the skills that are indicated on the same line as the corresponding checkbox. Team leaders are also able to delete skills directly from this screen and can edit or add new skills by selecting different links. Skills are slightly more complex than prerequisites and require a few more options, but they still require far less than a task (Figure 12).

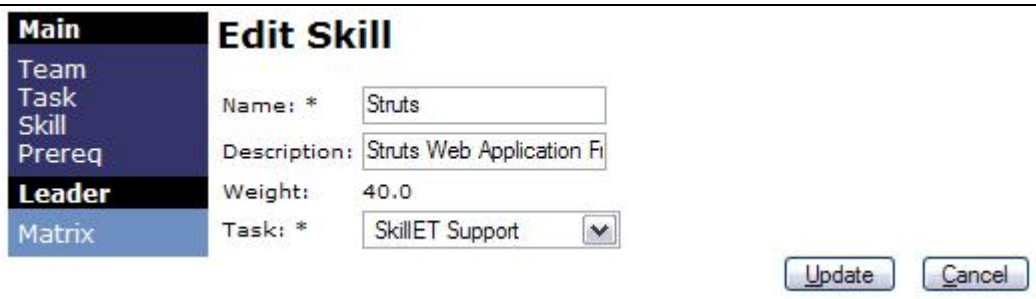


Figure 12 – Basic Edit Skills Screen

It is worthwhile to note that the task to which a skill belongs must already be created for that skill to be attached to at this time.

During development it was noted by business analysts that there was a small problem with the skills and prerequisites listing screens. These are the screens that both users and team leaders go to for updating which skills and prerequisites they possess. When the list of skills and prerequisites grew to more than one screen in length every update would bring the screen back to the top. This was happening because the forms on the page were doing traditional submits action and the page was being re-rendered to show the result. In order to combat this problem these pages were updated to again use AJAX to submit the data from these pages. This let the information be sent to the server but the page did not have to be updated afterwards. Only the sections of the page that had relevant changes were updated.

Once all of the prerequisites and skills for a task have been created, a team leader may go back and edit that task to properly assign all of the prerequisites and skills and also to update the weight of each of the skills on the task.

5.4 Security

In order to preserve data integrity in the application and to enhance users' trust of the application, security needed to be implemented. Security standards and guidelines are written and followed for all Web applications developed by the Application Development Consulting team. These standards include using NTLM authentication from any user's operating system login for authentication. Further, the user's Active Directory Security Groups will be used to determine what levels of access the user will have throughout the application. Each user's authentication is completed by redirecting the user to an intermediate ASP Web application which pulls the user's encrypted credentials from their browser. These credentials are then matched against the user's Active Directory profile

and their information is returned to the skillet application. Once this information is received the user's information is built into a secure business object that is placed in the user's session with the Web application server, BEA WebLogic in this case. Whenever the user attempts to access any page in the application that session object is checked for proper credentials before returning any response to the user. If the user's authentication of authorization fails at any point in the application for any reason they are redirected to a generic error page that displays a message stating why they were denied access.

5.5 Task Matrix

The initial impetus for the SkillET project came from a Cintas Six Sigma initiative. This program requires that graphical representations of quality are displayed for members of a team to see. For this reason two matrices can be generated by team leaders. This first one, which is used more often, would not be displayed for the whole team to see. It is used only for the team leader to get a quick glance at the team's flexibility. This first matrix gives a graphical representation of each team member's flexibility for each task as well as the team coverage for that task (Figure 13).

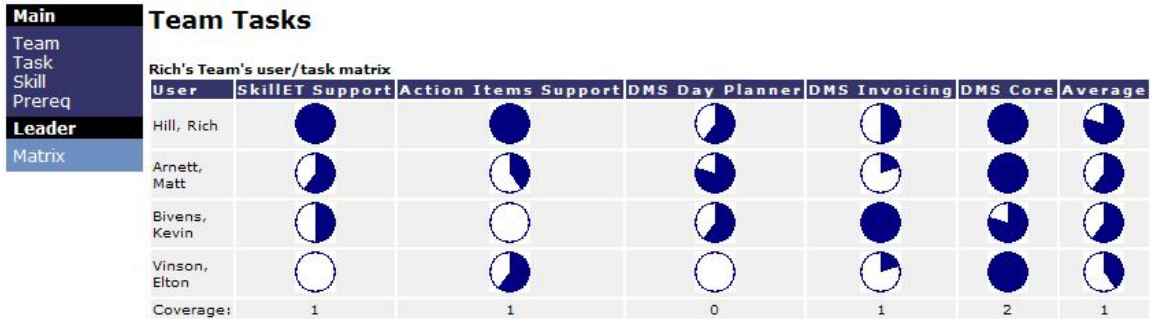


Figure 13 – Team Task Matrix with Users

The second matrix is merely a subset of the first matrix which shows only average team flexibility and task coverage (figure 14).

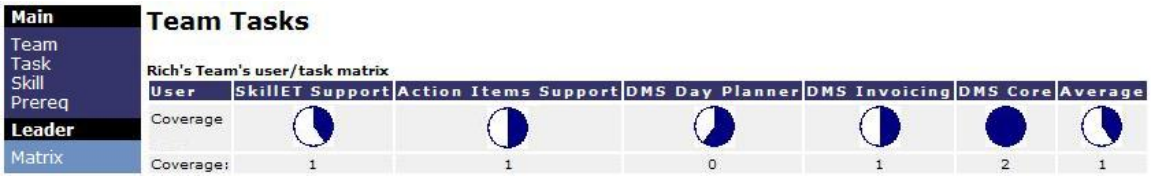


Figure 14 – Team Task Matrix without Users

Both of these matrices are reached by team leaders clicking on the Matrix link of the left menu bar. This link in the menu bar is only available to team leaders and the matrices are only available for teams that the manager leads. If a team leader manages more than one team then they must select which teams they wish to view the matrices for (Figure 15).

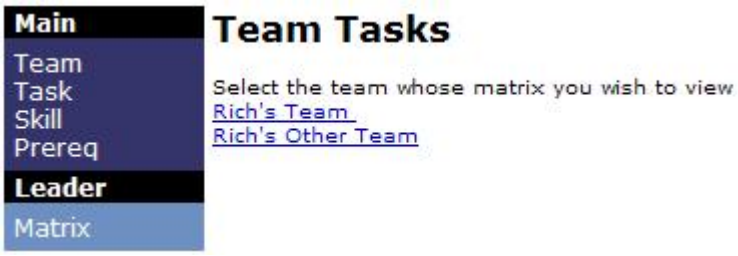


Figure 15 – Team Task Matrix Selection Screen

5.6 Excel Spreadsheet Generation

To allow team leaders to more easily manipulate the data in the SkillET application they would need to be able to massage that data. The standard inside of Cintas for data analysis of this type is to use Microsoft Excel©. Rather than have team leaders print out the data from the team task pages it was decided that we would automate the export of that data to Excel for them. To do this I decided to use the open source

project, Apache POI. Apache POI is allows Java developers to create a logical construct of a variety of Microsoft Office files and will then create the file to be used at the developers discretion. In this case I chose to generate the file and then send the resulting file to the user through the browser response object. In doing this I was able to avoid using an indeterminate amount of hard disk space in the future because the file is never saved to the server. If the user needs data for their team again in the future then they are simply able to click the same link and the data will be generated on the fly again.

Apache POI was not the easiest framework to use. Though the API documentation was substantial, no updates have been made to the project for several years. The performance for spreadsheet generation was poor, but this was considered to be okay from the perspective of the SkillET business analyst since this operation will be rarely performed.

5.7 SQL Server Backend using Hibernate

I was able to user a SQL Server 2000 database that was already present at Cintas to store the data used in the SkillET application. Creation of the database was very straightforward. That was not because the business objects were so simple, but rather that the SQL create scripts for the tables to be used in SkillET were automatically generated during each Ant build task for SkillET by using an extension of the Hibernate Object Relational Mapping (ORM) framework and the XDoclet code commenting tool. XDoclet was used to find specific code comments used by Hibernate. Hibernate then takes these comments and uses them to generate a SQL build script that will represent the business objects in a relational database. Hibernate was also able to use these same comments to generate XML files that it uses at run time to automatically persist these

objects to the database. The benefit of this is that hibernate handles the persistence and retrieval of all objects in and out of the database which relieves a lot of the standard busy work done by developers. Hibernate allowed me to build the entire SkillET application without having to manually manage a single connection or transaction and without having to write a single SQL command in code. The Hibernate framework is also extensible enough to allow developers to write custom SQL when it is needed, although it rarely is.

5.8 Designed to be flexible, scalable and verifiable

It is often easy to tout that a program has been well designed but it is difficult to prove it as such. In order to make the SkillET application more flexible I used a variety of frameworks that allow the application to be modified very quickly and very drastically with minimal effort. The Model – View – Controller pattern is used extensively throughout the application. The Struts Web application framework makes this easy by providing a simple way to isolate the view layer from the rest of the application. By taking this a step further I isolated all of the business objects to their own independent packages and enforced isolation of database access by only allowing persistence to be done through EJBs. Calling these EJBs as a separate service isolates the view layer from having to do anything with the model layer. Controllers are also handled by Struts in a safer manner because it manages the request and response objects from the client browser. Now that it is a bit clearer how each of the levels of the application are isolated the tricky part is showing how easy they are to change.

First, the model layer is isolated from the rest of the application by requiring access to be done through EJBs. This allows the other layers to expect a common

interface to use and the model layer can expect consistent input. Once the objects are received, they are given to the Hibernate framework which handles the persistence based solely on definitions of objects that are located in XML files. All that needs to be done to modify the model layer of the application is to change a few XML nodes and the entire persistence model could be changed. But just to make it a little easier, those XML files are generated by comments in the code. So developers can make sweeping application changes by adjusting comments and don't even have to worry about dipping into complex XML files.

Secondly, the controller layer of the application is all done inside of helper classes that manage business rules. These helper classes are called by the EJBs to ensure proper object flow and are also used by Struts action classes to act as an intermediary. Since all of the business rules are handled consistently and in an isolated area, they are easy to modify and maintain.

Finally, the view layer of the application is managed by the Struts Web application framework. Many of the common tasks that developers write over and over again are handled automatically. Simple to read and manage, XML files are used to handle input validation on both the client and server. The tiles plug-in for Struts allows developers to quickly change the layout of the entire application just by updating a template style Java Server Page (JSP). Further more, the tiles plug-in also allows developers to inject the results of any section of the application into a predefined area of the screen, giving developers the control of frames and the feel of a standard Web page.

As an additional, and easy to modify, part of the SkillET application, all information logging is handled using the Log4J framework which allows for a simple and

modifiable logging setup that is managed using property files. The amount and method of logging throughout the application can be managed through just one file containing name value pairs.

All of the previous items allow the application to be more flexible and scalable, but it was determined early on that there must be a way to verify that all of this enhanced architecture actually works. To make the application as easy to verify as possible I used the JUnit testing framework. Using JUnit in combination with the Emma code coverage API I was able to show deterministically that every line of executable code in the application was being tested. JUnit tests were run during each build of the application and Emma reports were generated to show how much of the code was being tested.

However, 100% testing coverage and testing during each build aren't hard to do if the tests are written to match the code that already runs. For this reason the Application Development Consulting team uses Test Driven Development (TDD). TDD is a subset of the Agile Programming methodology. TDD requires that developers write unit tests for their code before the code is written. This is done to ensure that there is code in place that matches business rules and requirements and also to ensure that code is not written for which there is no business value. TDD was used throughout the development of the SkillET application.

6. Testing Procedures

6.1 Unit Testing

As mentioned in the previous section unit testing was done prior to any code being written throughout this application. The JUnit framework was used extensively. JUnit by itself however was not enough to thoroughly test this application. In order to check that correct data was being persisted to the database for SkillET the DBUnit[1] API was used in conjunction with JUnit. DBUnit allows developers to define XML files that represent data located in the database. These files can then be used to either prepare data in the database or to compare to data that already exists to check for consistency.

Another framework that was used during unit testing for SkillET was Apache Cactus. Cactus allows developers to test service within the application. Cactus was necessary for testing the EJBs in SkillET and also to test the integration between the different layers of SkillET.

6.2 Acceptance Testing

The SkillET application server was first sent to a quality assurance server just three weeks into development. Access to this server and the SkillET application was given to the business analyst with whom I worked on the application. From that point forward the application was tested at least once per week by business users. Updates to SkillET were pushed out to the quality assurance server at least once per week so that new features could be tested and confirmed.

6.3 Regression Testing

Part of the job of the business analyst was to ensure that new features and functionality worked as expected. Beyond that, however, was the need to make sure those sections of the application that were previously working still did so. In order to do this the business analyst and I used Quick Test Professional by Mercury interactive to write scripts that could be executed against the application in a repeatable manner. This prevented the business analyst from having to check the entire application every week and would prove exceptionally useful if SkillET were to grow in scope.

6.4 Performance Testing

The last bit of testing for SkillET was to ensure that it was not using excessive resources to perform any of its tasks. To do this, Borland's OptimizeIt Server Trace was installed on both the test and quality assurance servers. From that point, if the application server instance was launched to track performance, I was able to load the Server Trace client and view how the code was being executed. Server Trace allowed me to ensure that there were no connection leaks occurring, that superfluous calls to methods were not being made and that no block of code was taking an inordinately long amount of time to complete. Several issues were detected by server trace that would never have shown up through other forms of testing and it allowed me to optimize performance in several key areas of the SkillET application.

7. Conclusions and Recommendations

7.1 Conclusions

The SkillET project was designed to help Cintas track the flexibility of its teams within the MIS department. It is a Web based application that will easily scale up to fit future needs. It allows team leaders to create new teams and to define their needs for flexibility. It allows the members of those teams to update their own skills and prerequisites to gain an accurate measurement of where the team's true flexibility is at. It provides robust reporting features and allows for team leaders to get a quick view of their teams' progress in a visual format. The SkillET application is fully tested and as such should be able to be updated in the future with little to no risk.

7.2 Recommendations

I encountered a few difficulties in creating this project. Mostly though, things went rather smoothly. Leverage existing frameworks reduced the amount of time required to build a solid project and increased the quality of the output. Knowing the resulting product of my hard work there are a few things I would go back and change.

The single most taxing part of the application was getting all of the code I developed to work properly with the frameworks I used and also getting the frameworks to behave well together. This is especially noticeable during testing when service locator objects and factory patterns have to be used to make use of mock objects or other testing conventions. If I were starting again at this point I would attempt to leverage and Inversion of Control (IoC) framework to ease the connection of different parts of the project. The Spring Dependency Injection framework implements IoC and allows for an

easier integration between disparate sections a project. I might also consider using the Spring MVC framework as opposed to the Struts Web application framework to build a greater degree of consistency into the project.

One of the late additions to my project was AJAX. I used AJAX to provide a richer user interface to the Web application. If I were starting again I would use this handy tool much more frequently, to the point that it was ubiquitous throughout the application. AJAX was simple to use and added a lot of benefit for very little extra effort. Unfortunately to modify all of the application by the time that I discovered how to use AJAX would have required major refactoring.

One item that I would have removed entirely from the application is the use of XDoclet. Though the technique of automatically generating XML files seemed interesting, it proved to be nothing more than a cute trick. I would rather learn the proper structure of the XML files used by Hibernate and manage them in a more precise manner than relying on an automatic code generation tool. XDoclet is not without merits, but I think learning the commenting syntax for it on top of learning the XML structure used by Hibernate is redundant. If one were to choose to only learn the XDoclet syntax then I would say that it was dangerous to put so much faith in two independent frameworks operating with such harmony.

There are however many things that I would not change about the way this project was developed. I think the use of frameworks added to the stability and functionality that I was able to produce. Taking advantage of validation, formatting, logging, request forwarding and persistence that has already been written and tested just makes good common sense. Why would anyone revert to writing all of that code that needs its own

testing and could be more error prone or less efficient than what is already available? In the future I plan to continue to take advantage of frameworks like Struts and Hibernate as they are the very essence of principled code reuse. The ability to reuse code that is not a part of the core language has been sorely missing from both the Java and .NET camps of programming. I for one would like to rejoice in making better projects in less time.

Appendices

A1. Integrating Frameworks – A code example

The bulk of the work in my application involves the use of several key frameworks. By using XDoclet, Struts, Struts Validator, Struts Tiles, Hibernate and AJAX I was able to quickly combine simple XML, code comments and a minimal amount of code to produce a robust feature set. Here I'll show a quick example of everything that's needed to complete the page which lists available skills. This page is also used to indicate which skills a user possess and updates task completion all without reloading the page.

First to be created is the class that is represented. In this case it is the Skill class. All of the functionality for creating, reading, updating and deleting skills in SkillET will ultimately be contained in the comments of this class. Note the special comments that begin @hibernate.

Skill.java

```
/**
 * @hibernate.class table="SKILLET_SKILLS"
 */
public class Skill
{
  private int id;
  private String name;
  private String description;
  private Task task;
  private float weight;
  private List users;

  /**
   * @hibernate.id column="skillId" type="integer" generator-class="native" unsaved-value="-1"
   */
  public int getId()
  {
    return id;
  }

  public void setId(int id)
```

```
  {
    this.id = id;
  }

…
…
…

  /**
   * @hibernate.many-to-one column="taskId" class="com.cintas.skillet.bus.Task"not-null="true"
   */
  public Task getTask()
  {
    return task;
  }

  public void setTask(Task task)
  {
    this.task = task;
  }
…
…
…
}
```

An Ant build script is used to compile and deploy SkillET each time it is ready to be rolled out. The Ant script is configurable from the command line to do as much or as little as is needed for each build. For SkillET, a single Ant task named “all” is used to compile the code, format the code to Cintas standards, generating an XML file to represent each class with our special comments for Hibernate, generate a SQL script to build a database that matches those XML files and deploy the application to the WebLogic servers that are used to host SkillET.

build.xml

```
<?xml version="1.0"?>
<project name="Cintas SkillET Project" default="all" basedir=".">

  <description>
    The Cintas skillet application ANT build script
  </description>

  <property name="scm" location="../../.." />
  <property name="import.dir" location="${scm}/Standards/Java/Frameworks/build" />
  <property name="tools" location="${scm}/Standards/Java/Tools" />
```

```
    <!-- Do Everything -->
    <target name="all" depends="dist,deploy" description="Default task: Build distribution."/>

    <!—Format all source code -->
    <target name="format" description="Formats code to Cintas standards">
      <jalopy convention="${jalopy.config}" loglevel="warn" classpathref="class.path"
            history="file" historymethod="adler32">
        <fileset dir="${src.java}">
          <patternset refid="source.set"/>
        </fileset>
        <fileset dir="${src.test}">
          <patternset refid="source.set"/>
        </fileset>
      </jalopy>
    </target>

    <!-- Make hibernate files using XDoclet -->
    <target name="hib-xdoclet" depends="compile" description="Generates Hibernate class
descriptor files">
      <taskdef name="hibernatedoclet"
classname="xdoclet.modules.hibernate.HibernateDocletTask">
        <classpath>
          <path refid="xdoclet.path"/>
          <path refid="framework.class.path"/>
        </classpath>
      </taskdef>

      <hibernatedoclet destDir="${classes}" excludedtags="@version,@author,@todo" force="true"
verbose="false">
          <fileset dir="${src.java}">
            <include name="**/*.java"/>
          </fileset>
        <hibernate version="2.0"/>
      </hibernatedoclet>
    </target>

    <!-- Generate a SQL file that can create that database for Hibernate -->
    <target name="hib-schema" depends="hib-xdoclet" description="Generates the schema that is
defined by all the Hibernate descriptor files.">
      <taskdef name="schemaexport"
classname="net.sf.hibernate.tool.hbm2ddl.SchemaExportTask">
        <classpath>
          <path refid="hibernate.path"/>
          <path refid="framework.class.path"/>
          <path refid="commons.class.path"/>
          <pathelement location="${classes}"/>
        </classpath>
      </taskdef>

   …
   …
   …

    <target name="ear" depends="ejb,war,ear-compile" description="Creates a deployable EAR
package."/>
```

```
  <target name="dist" depends="build-version,ear"
        description="Builds a distributable versions of the application." />

  <target name="dist-init">
    <mkdir dir="${dist}" />
  </target>

  <target name="dist-clean" depends="ear-clean,war-clean,jar-clean">
    <!--delete dir="${dist}" /-->
  </target>

  <target name="deploy" description="Deploy application EAR.">
    <wldeploy user="*****"
            password="*****"
            adminurl="*****"
            action="deploy"
            name="skillet"
            targets="*****" />
  </target>
</project>
```

XDoclet was used during the Ant build target “hib-xdoclet” to generate the Skill.hbm.xml file used by Hibernate to persist Skill objects to the database and to retrieve them. The structure of the XML file is defined by a Data Type Definition (DTD) file. This ensures that the contents of the XML file will be consistent and proper for hibernate to use. This file contains all of the information Hibernate needs to manage the Skill class. This partial version of the file tells Hibernate how to manage the id and task attributes of the class.

Skill.hbm.xml

```
<?xml version="1.0"?>

<!DOCTYPE hibernate-mapping PUBLIC
   "-//Hibernate/Hibernate Mapping DTD 2.0//EN"
   "http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd">

<hibernate-mapping>
   <class name="com.cintas.skillet.bus.Skill"
      table="SKILLET_SKILLS"
      dynamic-update="false"
      dynamic-insert="false">

      <id name="id"
         column="skillId"
         type="integer"
```

```
            unsaved-value="-1" >

            <generator class="native"/>
        </id>

    …
    …
    …

        <many-to-one
            name="task"
            class="com.cintas.skillet.bus.Task"
            cascade="none"
            outer-join="auto"
            update="true"
            insert="true"
            access="property"
            column="taskId"
            not-null="true"
        />
    </class>
</hibernate-mapping>
```

Hibernate actually only uses one XML file, hibernate.xml. This file will just contain references to the XML files that are generated during the build process. For the purposes of this appendix it is irrelevant.

The SkillET deployment descriptor contains a filter that determines how all requests are handled. In this case, any requests to this application where the uniform resource locator (URL) ends in ".do" will be forwarded to the Struts Web application framework for further processing.

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <display-name>skillet</display-name>
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
    <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
```

```
    <init-param>
      <param-name>application</param-name>
      <param-value>ApplicationResources</param-value>
    </init-param>
    <load-on-startup>2</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>

…
…
…

</web-app>
```

Struts uses the file struts-config.xml, defined in the web.xml file, to process all requests that the framework receives. In this case, the request is to skill.do with a GET parameter of “Action” equal to “listSkills”. Struts interprets this request by looking for a pattern in the configuration file that matches the input URL. The tag “actionName” is defined as a parameter for this Struts action which will be significant shortly. The struts-config.xml file entry for this action also points to a specific Struts action class that is used for processing. Here the action class is com.cintas.skillet.bus.SkillAction. This means that this request will be forwarded to that action class. The struts-config.xml file also contains references that load in the validator and tiles plug-ins.

struts-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts Configuration
1.1//EN" "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">

<struts-config>
  <form-beans>
    <form-bean name="SkillForm" type="com.cintas.skillet.web.forms.SkillForm" />
  </form-beans>

  <action-mappings>

    <action name="SkillForm" parameter="actionName" path="/skills" scope="request"
type="com.cintas.skillet.web.actions.SkillAction" validate="false">
      <forward name="listSkills" path="listSkills" />
```

```
    </action>

  </action-mappings>

  <plug-in className="org.apache.struts.tiles.TilesPlugin">
    <set-property property="definitions-config" value="/WEB-INF/tiles-defs.xml" />
  </plug-in>

  <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
    <set-property property="pathnames" value="/WEB-INF/validator-rules.xml,/WEB-
INF/validation.xml" />
  </plug-in>

</struts-config>
```

The SkillAction class extends the DispatchAction which is included in the Struts framework. This class is used by Struts to determine what action is taken when the request is forwarded by the framework. Since the SkillAction class extends the DispatchAction class reflection is used to determine which method is executed at run time. In this case the struts-config.xml file entry for "skills" listed actionName as its parameter. That means that if there is a request parameter named actionName in the request then the value of that parameter is the name of the method to be executed. In this case the parameter was set to "listSkills" so the method named listSkills will be executed.

SkillAction.java

```
package com.cintas.skillet.web.actions;

import java.util.*;
import javax.servlet.http.*;
import org.apache.commons.lang.StringUtils;
import org.apache.struts.action.*;
import org.apache.struts.actions.DispatchAction;
import com.cintas.skillet.bus.*;
import com.cintas.skillet.services.ServicesLocator;
import com.cintas.skillet.services.skill.SkillServiceLocal;
import com.cintas.util.ValidateException;

public class SkillAction extends DispatchAction
{
  public ActionForward listSkills(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response)
    throws Exception
  {
```

```
    String forward = "listSkills";
    Map userSkills = new HashMap();
    Map userFlexibility = new HashMap();
    List skills = new ArrayList();
    List teams = null;
    List leadTeams = null;
    User user = null;

…
…
…

    skills = user.getSkills();
    if(skills != null)
    {
      for(int i = 0; i < skills.size(); i++)
      {
        userSkills.put(skills.get(i), "1");
      }
    }

    if(user.getTeams() != null)
    {
      teams = user.getTeams();

      for(int i = 0; i < teams.size(); i++)
      {
        Team team = (Team) teams.get(i);

        if(team.getTasks() != null)
        {
          List tasks = team.getTasks();

          for(int j = 0; j < tasks.size(); j++)
          {
            Task task = (Task) tasks.get(j);

            userFlexibility.put(task, Float.toString(user.getFlexibility(task)));
          }
        }
      }
    }

…
…
…

    request.setAttribute("userSkills", userSkills);
    request.setAttribute("userFlexibility", userFlexibility);
    teams = user.getTeams();
    leadTeams = user.getLeadTeams();
    request.setAttribute("teams", teams);
    request.setAttribute("leadTeams", leadTeams);

    return mapping.findForward(forward);
  }
```

```
  public ActionForward storeSkill(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response)
    throws Exception
  {
    SkillServiceLocal skillsl =ServicesLocator.getInstance().getSkillServiceLocalHome().create();
    Skill skill = new Skill();
    SkillForm sf = (SkillForm) form;
    int taskId = 0;

    skill.setId(Integer.parseInt(sf.getId()));
    skill.setName(sf.getName());
    skill.setDescription(sf.getDescription());
    skill.setWeight(sf.getWeight());

…
…
…

    if(taskId > 0)
    {
      TaskServiceLocal tsl = ServicesLocator.getInstance().getTaskServiceLocalHome().create();
      skill.setTask(tsl.findTaskById(taskId));
    }
    else
    {
      throw new ValidateException("Invalid task associated with this skill",
        new String[] { "taskId" });
    }

    if(skill.getId() == 0)
    {
      skill.setId(-1);
    }

    skillsl.store(skill);
    return listSkills(mapping, form, request, response);
  }
}
```

The return value from the listSkills method is of type ActionForward. Struts uses that ActionForward to determine where the next step in the request process goes. Looking back at the struts-config.xml file the forward goes to listSkills. listSkills is actually a reference to the tiles plug-in for Struts. Tiles is a module for struts that allows a template based design that is easily modifiable. The struts-config.xml file defines the tiles definitions to be located in a file called tiles-defs.xml. The tiles-defs.xml file defines

how pages are rendered for the user. It takes a composite of many JSP files and renders them as a single response page to the user. The listSkills definition extends the masterLayout definition which is used in SkillET as the base design for all pages. By giving the masterLayout template a series of inputs the page is rendered. In this case everything stays the same on the masterLayout except the page title and the contents of the body section of the page. This type of layout is very similar to standard html frames or JSP insert statements.

tiles-defs.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE tiles-definitions PUBLIC "-//Apache Software Foundation//DTD Tiles Configuration 1.1//EN" "http://jakarta.apache.org/struts/dtds/tiles-config_1_1.dtd">

<tiles-definitions>
  <definition name="masterLayout" path="/WEB-INF/jsp/layout/defaultLayout.jsp">
    <put name="title" value="Cintas SkillET" />
    <put name="header.title" value="Cintas SkillET" />
    <put name="header" value="/WEB-INF/jsp/common/header.jsp" />
    <put name="mainMenu" value="/WEB-INF/jsp/menus/mainMenu.jsp" />
    <put name="leaderMenu" value="/WEB-INF/jsp/menus/leaderMenu.jsp" />
    <put name="body-content" value="/WEB-INF/jsp/content/home.jsp" />
    <put name="error-content" value="/WEB-INF/jsp/exceptions/validateerror.jsp" />
    <put name="footer" value="/WEB-INF/jsp/common/footer.jsp" />
  </definition>

…
…
…

  <definition extends="masterLayout" name="listSkills">
    <put name="header.title" value="Skills" />
    <put name="body-content" value="/WEB-INF/jsp/content/skill/listSkills.jsp" />
  </definition>
  <definition extends="masterLayout" name="editSkill">
    <put name="header.title" value="Edit Skill" />
    <put name="body-content" value="/WEB-INF/jsp/content/skill/editSkill.jsp" />
  </definition>
</tiles-definitions>
```

A quick look at the defaultLayout.jsp file which is used by masterLayout will give an idea of how the final page will look. You can see where the header.title and body-content sections are defined as they will be overridden when rendering the final page.

defaultLayout.jsp

```
<%@ page contentType="text/html" pageEncoding="ISO-8859-1" %>
<%@ taglib uri="struts-html" prefix="html"%>
<%@ taglib uri="struts-tiles" prefix="tiles"%>
<%@ taglib uri="jstl-core" prefix="c" %>


<html:html>

<head>
  <title><tiles:getAsString name="title"/></title>
  <link rel="stylesheet" type="text/css" href="/skillet/style/skilletStyle.css"/>
  <script src="/skillet/scriptFiles/header.js"></script>
  <script src="/skillet/scriptFiles/math.js"></script>
</head>


<body topmargin="0" leftmargin="0" onload="setFocus('text','textarea')">

  <script type="text/javascript">
    var submitCount = 0;
  </script>

  <!-- header -->
  <table>
    <tr>
      <td width="800">
        <tiles:insert attribute="header"/>
      </td>
    </tr>
  </table>
  <!-- end of header -->

  <!-- middle section (menu bar and body content) -->
  <table width="800">
    <tr>

      <!-- Menu bar -->
      <td width="100" align="left" valign="top">
        <tiles:insert attribute="mainMenu"/>
        <c:if test='${sessionScope.IS_ADMIN eq "true"}'>
          <tiles:insert attribute="leaderMenu"/>
        </c:if>
        <img src="/skillet/images/menuTorsionBar.gif" border="0" width="100"/>
      </td>

      <!-- Main body information -->
      <td align="left" valign="top">
        <h1 class="title">
          <tiles:getAsString name="header.title"/>
        </h1>
        <tiles:insert attribute="error-content"/>
        <tiles:insert attribute="body-content"/>
      </td>
```

```
    </tr>
  </table>
  <!-- end of main body -->

  <!-- footer -->
  <table>
    <tr>
      <td width="800">
        <tiles:insert attribute="footer" />
      </td>
    <tr>
  </table>
  <!-- end of footer -->

</body>
</html:html>
```

The tiles-defs.xml file defined the body section as being overridden by the listSkills.jsp file. This is a standard JSP file that uses some server side processing on the objects that were added to the request from the SkillAction class earlier. This makes putting objects on the page very simple. Simple looping using the Java Standard Tag Library (JSTL) allows me to easily iterate over a collection of objects, in this case teams and skills. There is also a JSTL tag used to format numeric output to the user which makes sending a floating point value through the request and outputting it in a user friendly way a trivial task. Special Struts tags, as defined in tag libraries listed in the web.xml deployment descriptor, are used throughout the page to allow Struts to manage many of the objects on the page and to make processing easier.

listSkills.jsp

```
<%@ taglib uri="jstl-core" prefix="c" %>
<%@ taglib uri="jstl-fmt" prefix="fmt" %>
<%@ taglib uri="struts-html" prefix="html" %>

<html:form action="skills.do?actionName=updateUserSkills">
  <html:hidden property="id"/>

…
…
…
```

```
  <c:forEach items="${leadTeams}" var="team">
    <table border="0" width="100%">
      <tr>
        <td align="left">
          <strong>
            <c:out value="${team.teamName}"/>'s Skills
          </strong>
        </td>
        <td align="right">
          <a href="skills.do?actionName=getSkill">
            <img alt="Add a new skill" src="images/plus.gif" border="0" />
            Add a new skill
          </a>
        </td>
      </tr>
    </table>
    <table width="100%" border="0">
      <tr class="heading">
        <td colspan="2">
          name
        </td>
        <td width="5%">
          edit
        </td>
        <td width="5%">
          delete
        </td>
        <td width="5%">
          learned
        </td>
      </tr>
      <c:forEach items="${team.tasks}" var="task">
        <tr class="row">
          <td align="left" colspan="4">
            <c:out value="${task.name}"/>
          </td>
          <td align="center">
            <fmt:formatNumber maxIntegerDigits="3" minIntegerDigits="1" maxFractionDigits="2"
minFractionDigits="2" value="${userFlexibility[task]}"/>
          </td>
        </tr>
        <c:forEach items="${task.skills}" var="skill">
          <tr class="row">
            <td width="5%">
               
            </td>
            <td align="left">
              <c:out value="${skill.name}"/>
            </td>
            <td width="5%" align="center">
              <a href="skills.do?actionName=getSkill&skillId=<c:out value="${skill.id}"/>">
                <img alt="Edit this skill" src="images/edit.gif" border="0" />
              </a>
            </td>
            <td width="5%" align="center">
```

```
            <a href="skills.do?actionName=deleteSkill&skillId=<c:out value="${skill.id}"/>"
onClick="return confirmDelete()">
              <img alt="Delete this skill" src="images/delete.gif" border="0" />
            </a>
          </td>
          <td width="5%" align="center">
            <input type="checkbox" onclick="submitFormWithData( this.form, 'id', '<c:out
value="${skill.id}"/>' );"
              <c:if test="${userSkills[skill] eq '1'}">
                checked="checked"
              </c:if>
            />
          </td>
        </tr>
      </c:forEach>
    </c:forEach>
  </table>
  <br/>
 </c:forEach>

 </html:form>
```

At this point all of the processing is complete. The Struts tiles plug-in will assemble all of the pieces of the page together and return the result to the user. The finished product looks like Figure 16.

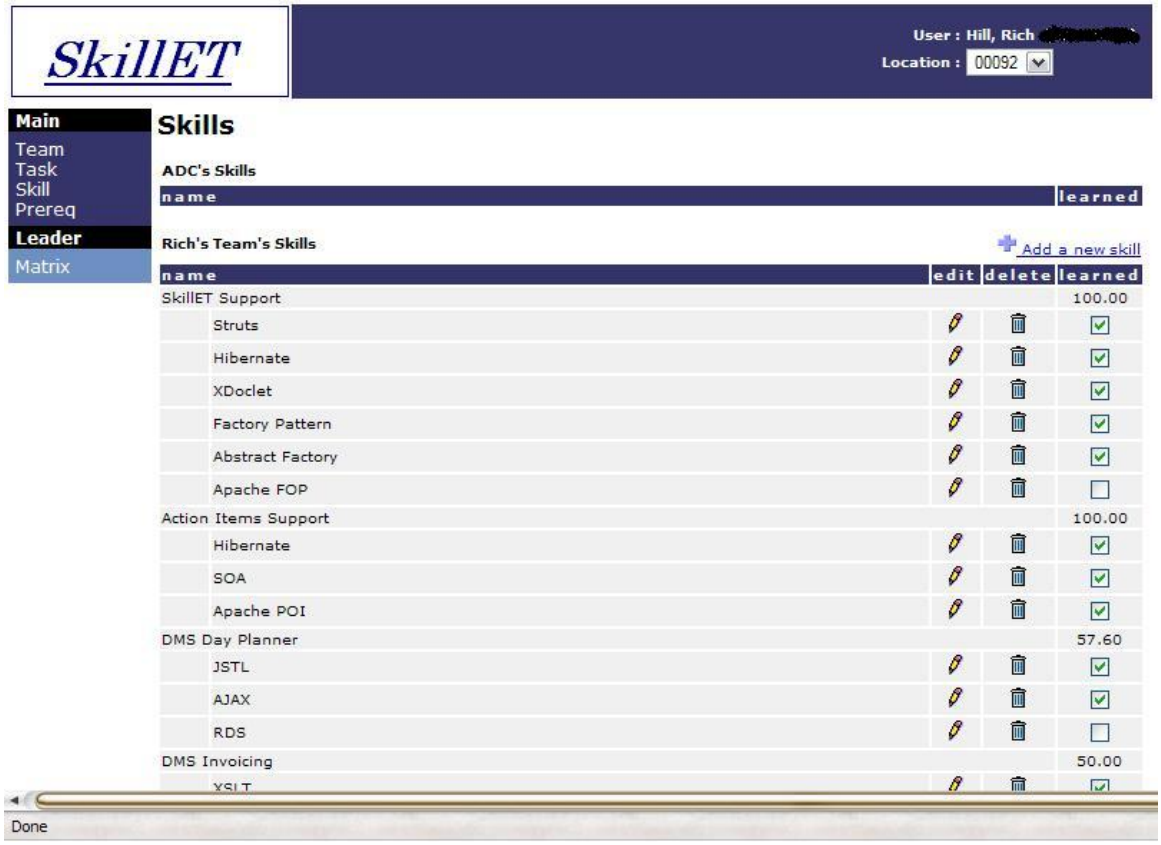


Figure 16 – List Users Final Rendering

References

1. DB Unit. “DBUnit, About DB Unit”. dbunit.sourceforge.net/ June 2, 2005
2. Cactus. “Jakarta Cactus”.jakarta.apache.org/cactus/ June 2, 2005
3. “Hibernate Object Relational Mapping Framework”. www.hibernate.org June 2, 2005
4. JUnit. “JUnit, Testing resources for Extreme Programming” www.junit.org/index.htm June 2, 2005
5. IsSixSigma Group. “What is six sigma?”. http://www.isixsigma.com/dictionary/5_Laws_of_Lean_Six_Sigma-663.htm November 2, 2004
6. Struts. “The Apache Struts Web Application Framework”. struts.apache.org/ June 2, 2005
7. XDoclet. “XDoclet – Attribute Oriented Programming”. xdoclet.sourceforge.net/xdoclet/index.html June 2, 2005