Android in the Real World

Roger Blumin
rblumin@live.com
GSET

Cory Brzycki
supercclank@gmail.com
GSET

Robert Hummel
rhummel314@aol.com
GSET

John Morone
fun498@gmail.com
GSET

Kimberly Son
kimberlyson@hotmail.com
GSET

New Jersey Governor's School of Engineering & Technology 2012

1 Abstract

After the telecommunications company Ericsson acquired Telcordia, data needed to be transferred from company to company. One specific instance was the long list of incoming processes that would confuse most managers. Technicians understand dense data and chaotic charts, but managers do not. Currently, many companies like Ericsson compile thousands of processes a day that managers must oversee, but the methods of presenting information are difficult to interpret and consume too much time. To make analyzing the data simpler, more efficient, and user-friendly, we created an Android application for Ericsson that presents the data in an organized manner. Graphs and charts proved to be effective ways to display the data and enable the user to quickly make well-informed decisions.

2 Introduction

To effectively convey live and archived business data in a meaningful and useful manner, three major design challenges had to be overcome: a natural Graphical User Interface (GUI), an efficient and logical Android architecture, and a concise and universal server protocol. These three challenges became the three main goals and pillars of design for the application. Keeping the problem and end user in mind, a purpose was defined from these pillars. This purpose served as a general guideline for all of the design decisions: clean, beautiful, fast, resource efficient, and universal.

2.1 Android

Android is a Linux-based operating system that runs on an open source platform. The Android Open Source Project, led by Google, was created to enable users to customize their phones to their needs and makes developing applications more convenient. Each updated version of android is given a dessert name in alphabetical order, Jellybean being the newest one [2]. The application was designed with Ice Cream Sandwich (second newest version) as the intended version, but it will be compatible for all tablets.

Android is primarily coded using Java in Eclipse with the Android (SDK) plug-in. Because Java is a multi-platform language, benefits of using Java include portability, object-oriented programming, and the ability to function on both the windows and Mac OS operating systems.

Object-oriented programming focuses on having multiple data structures that interact to form a complete application. It encourages organization, and isolation of data structures. Keeping parts of an application separate allows a programmer to limit certain data to only being able to be used internally. For example, if a user was granted direct access to a stored database, they could inadvertently corrupt important

files. Object-oriented programming is centered on the ideas of “objects” and “classes”. Classes are like the blueprints for “objects”, such as a house. Objects are physical instances of classes that can store and hold data. Many objects, like houses, can be created from the same general blueprint. It is the value of its variables that makes each object unique. The program created for Ericsson uses a class system to separate the server code from the User Interface.

The Android platform in Eclipse uses Extensible Markup Language (XML) for layout design, which is similar to HyperText Markup Language (HTML). Both languages can be used to code web pages, but XML is more flexible because it allows programmers to define their own scripting tags [3]. The application sends an XML request that communicates with Ericsson’s web services that in turn contacts the Oracle Database.

Business Process Management (BPM) was a central focus of the project as its purpose was building up a framework and an application to supplement and enhance Ericsson’s current desktop BPM programs. BPM is a method for organizing a business into a series of repeatable and reliable processes. BPM involves connecting people and computer systems to specific useful functions to create steps. These steps work together to form a process, which transforms a client request into a helpful response. Furthermore, these processes can be chained together to form more advanced processes, effectively turning the smaller processes into steps of the larger process, the prior processes into clients, and the subsequent processes into outputs. BPM is an approach to business engineering, and therefore targets maximizing overall efficiency and value through quantitative analysis and models. While these are concepts very familiar to engineers, they are usually much more esoteric to business managers. However, when there is a problem with a process, computer system, or employee, it is the manager that must make the decision, not a technician or engineer. As advanced as BPM has come in the past years, there is still a need to make the technical side of business understandable and accessible to those responsible for decisions [4].

2.2 SOAP-Simple Object Access Protocol

Simple Object Access Protocol (SOAP) refers to the protocol that is usually used in a Web service requests and responses to encode the information prior to sending it [5]. Through SOAP messages, XML code is exchanged between the client and server. SOAP allows a certain program from a computer to communicate with another program regardless of any differences in operating system. SOAP is useful because it is likely to bypass firewall, which allows programs to communicate with each [6]. Since SOAP messages cannot be composed directly on the Android platform, a third party library is necessary to allow an application to make contact with a server. KSOAP2 is a library that is designed to allow Android applications to compose and send SOAP messages. KSOAP2 also gives the programmer the ability to define how the messages are processed after they are received. This is what was used in our application in order to link to Ericsson’s server.

2.3 Servers

A significant portion of the project was communicating with Ericsson’s server to feed the application real time information. From the user’s Android device, a request is sent to the web service client in XML form. KSOAP2 sends the request to the database

which in turn sends a response and the XML is sent back to android and used in GUI. Oracle is used as a relational database management system in this project. It stores files, records, and other information in a database that can be accessed by numerous users and devices. Java is used to create a SOAP message written in XML, which is sent to the Android-BPM Service module. This module is responsible for interpreting the SOAP messages sent by the Android application and querying the server with the given request. The server's response is processed by the Service module, before returning back to the Android platform in XML.

3 "Software Design"

3.1 Visual/Organization/Presentation

This project required research about data visualization and the most effective way to communicate very complex data. A large focus of the early stages of development was the analysis of data contained within a test bed, or simulated server, provided by Ericsson. Once the data and its importance were fully understood, design began for creating an optimal layout that would be both user-friendly and informative. To ensure that all users would be able to comfortably use the application, there is a large emphasis on its appearance and accessibility. Among the focuses in regards to visual design was making sure the application would be accessible to users who are colorblind.

Research about color blindness was conducted using Color Oracle, a free software tool that simulates colorblindness. [7]. It changes the colors on the computer screen to colors that colorblind people are able to distinguish, depending on their type of colorblindness. One early idea was to create an option screen where the user could choose to run the application in a colorblind mode. However, since the main objective was to make complex information easily accessible and communicable, creating a universal color scheme was a better choice. The color scheme used is comprised of colors that are distinguishable to all users, regardless of the type of colorblindness. For instance, those with deuteranopia will see varying shades of green and blue. Users who do not have deuteranopia but cannot distinguish shades will see the screens in the regular colors that non-colorblind users see [7]. Not only did the colors have to be distinguishable from each other, but they also had to be universally symbolic. The palette ultimately used ranges from the universal negative (red) for aborted processes to the universal positive (blue) for active processes. Psychologically, red attracts attention and blue has a calming effect.

Another concern with accessibility was making sure that the user would be able to quickly distinguish crucial information from the rest of the displayed data. Enabling multiple preferences or settings would complicate the application and prove counterproductive to the original intention of the project: simplicity.

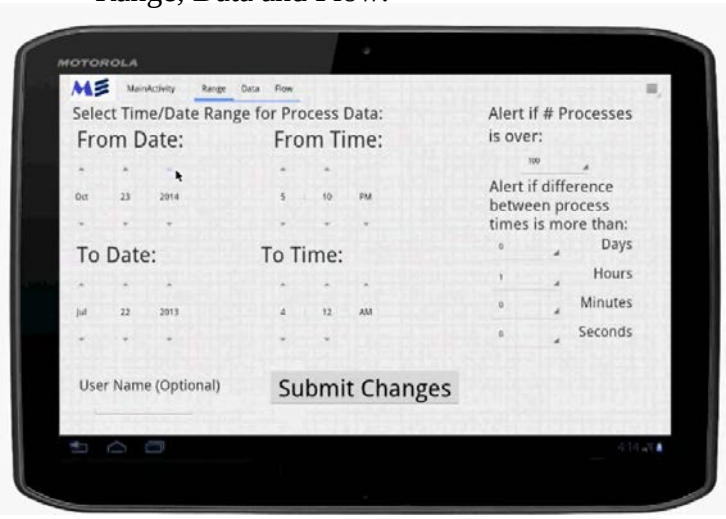
3.2 Development Strategies

Since graphic design was so integral to the purpose of the application, editing software such as Adobe Photoshop was used heavily in the development process. This allowed for quick changes during presentations with the company and in brainstorming GUI design. Throughout the infancy of the design process, the official Android design "building blocks" page was used for guidance [8]. It was essential that the screens proposed were feasible to render in our finished application. After creating a foundation to build upon, the Eclipse Development Environment was used to create the layouts for the project. Its large

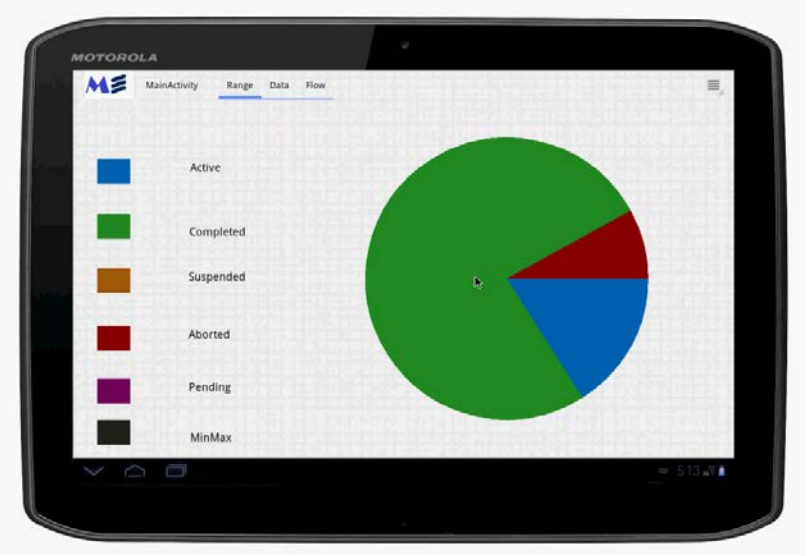
array of internal development tools and plug-in support heavily influenced the decision to use it. Included in these tools is a Package explorer menu that made organizing and combining individual source code files very efficient. Eclipse's support of external plug-ins allowed the easy installation of the Android Software Development Kit and create virtual Android Devices to test our code. Once the software development environment was correctly configured, the next step in software design was divided into two main components: User Interface and Server Contact development.

3.3 Graphical User Interface (See appendix A)

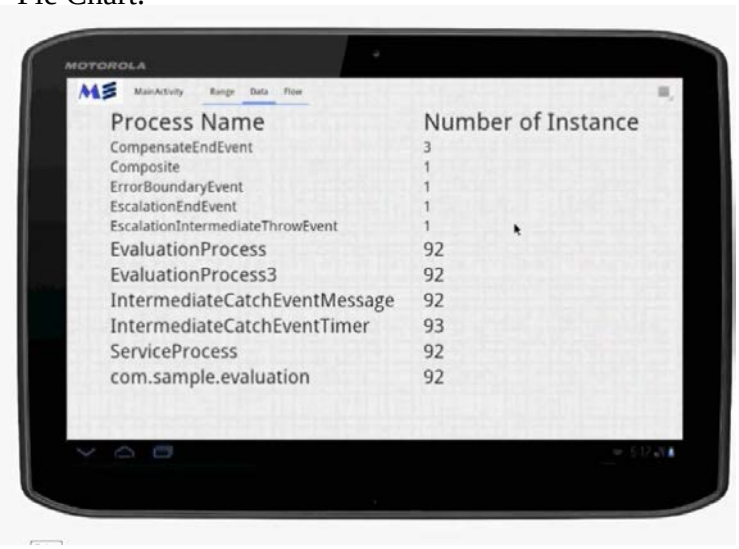
Most of the project team was tasked with designing the GUI, the aspect of the application that the user would be interacting with. It was decided that the user would navigate through three main screens: Range, Data and Flow.



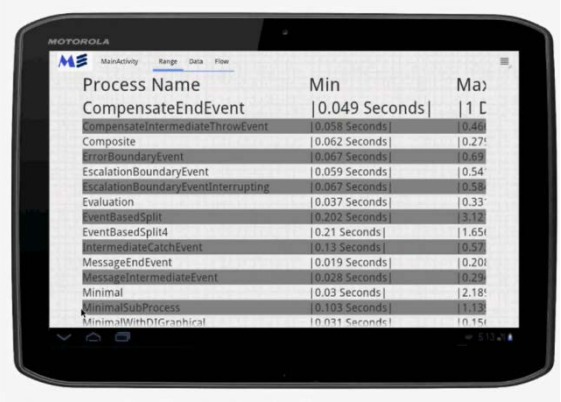
The range screen is where the user is able to set the date and time range of the data that they will be viewing in the other parts of the application. In the data screen, the user will first be presented with an overall view of the data for their selected date and time range in the form of a pie chart.



From this main data screen, the user will be able to get further information for each process status by clicking on the status they wish to focus on in the legend of the Pie Chart.



The next screen that the user will see after clicking on part of the Pie Chart legend is a list of processes that belong to a given status and the number of instances for each process.



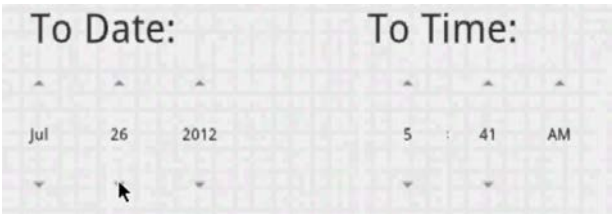
From the Pie Chart the user is also able to navigate to a separate min-max table that shows the minimum and maximum time that it has taken for each "Completed Process". The Flow screen shows a visual representation of the data in the form of a BPM Process Diagram for the given Process Flow. There is also an information screen that can be navigated to, using the dropdown menu in the action bar that provides the user with information about our application and Ericsson Incorporated.

3.4 Server

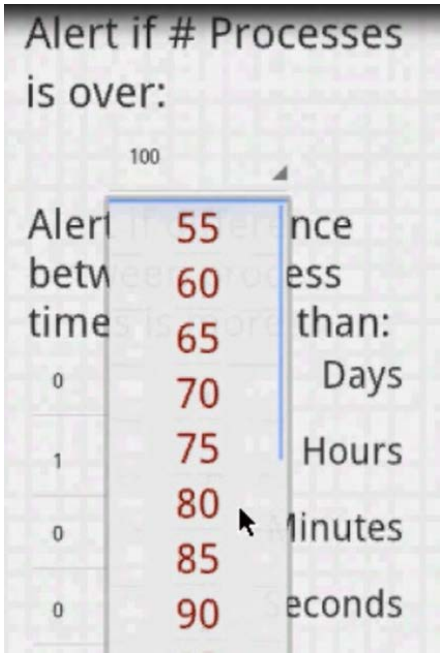
The remaining members of the team were given the challenge of finding an efficient way to fetch data from Ericsson's test bed server. This proved to be particularly difficult because the Android platform is limited and does not natively support the traditional ways that Java applications interact with servers. Various WebServices were experimented with in this stage of development. The first web accessor investigated was Java Architecture for XML Binding (JAXB) [9]. JAXB could not be implemented because Android does not support the JAXB application programming interface (API) included in Java. The next option was to investigate the Java API for XML WebServices (JAX-WS) and SOAP messages to determine whether or not it was supported by the Android Platform. However, standard JAX-WS was not native to android, but a third party library would allow the application to access the server. The library used, KSOAP2, is a registered open source tool designed specifically to aid developers who need to contact servers in their Android Applications. Next, class files had to be created to implement the methods contained in the library to send a request to the server and process the received response [5]. During this period of experimenting with KSOAP2, Transmission Control Protocol Monitor (TCPMON), another open source tool, was used to monitor and examine the XML messages being exchanged between the application and the server [10].

3.5 Display

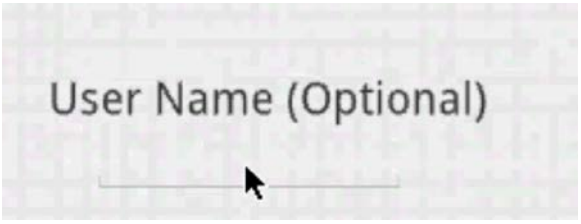
The first tab that appears when the user starts the application uses DatePickers, TimePickers, and Spinners as a means of acquiring data from the user. These are form elements native to Android. TimePickers are displayed as a set of three numbers the user can scroll through – hours, minutes, and AM/PM.



The DatePicker shows up the same way, only it displays the month, day, and year as three numbers to scroll through. Spinners are basically dropdown menus. In the application, they are used to set alert thresholds.



The user specifies what he or she deems as a large number of instances for a particular process, and what range is large enough to require an alert. The defaults for the application look at the system clock and set the process range to one hour prior, since most of the managers who use the app will want current information. The default threshold for number of instances of a process is 100, and the default difference between process times to cause concern is one hour. By changing these values, the user can acquire any amount of information from any time during the company's existence.
This main input screen also has a textbox for a user name.

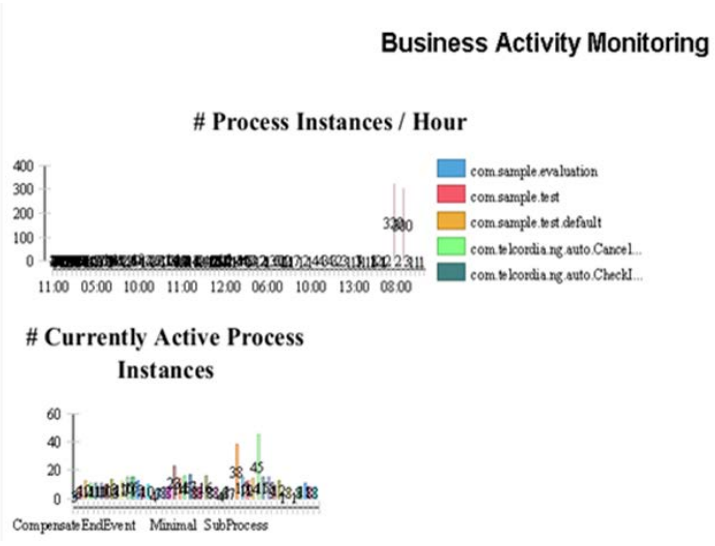


If this field is utilized, the functions that call the server will filter the processes to only ones that this person started. If left blank, it simply returns all of them. Several other spinners are also on the form, to set the thresholds of difference for the processes to be highlighted. The second tab, which displays the pie chart of processes with different statuses, links to dynamic tables. These tables list the computer processes that fired within the threshold of time that the user set previously. If the completed processes are selected, the user is directed to a table of minimum and maximum times taken from the server. The application uses a method to check the longest the process took against the minimum amount of time it took to complete a process. If the difference between the shortest and longest instance exceeds the difference set on the other screen, the entire row increases in font size. Similarly, clicking any of the buttons on the key for the pie chart will take the user to a page that lists the number of instances of a process next to their names. If the number of times a process was initialized exceeds the threshold the user set earlier, it is increased in font size to draw attention.

The Spinners, TimePickers, and DatePickers have native methods to get the values for each of the selected parts, such as the year, month, seconds, etc. These methods are called in the application and concatenated into a string to be tokenized later for the server code. The years, months, and days are separated by dashes while the hours, minutes, and seconds are separated by colons. The dollar sign was established as the delimiting character to separate the strings that contain the "to" and "from" dates and times. The range setting screen recorded data (called "instance data" for the created class) earlier in the application. All of the variables that hold this data are declared as static because there is only one copy of them to be referenced in the program. This instance data is referenced to call the server in the creation of the tables. Its final format looks like this: yyyy-MM-dd$HH:mm:ss$yyyy-dd-MM$HH:mm:ss.
The progression is the start date, followed by start time, then end date, then end time.
The data returned from the server is in the same format, and needs to be interpreted into numbers instead of a large chunk of

text. This is done using Tokenizers. A String Tokenizer is a class native to Java. Its methods allow the separation of words or phrases by delimiting characters. We used one tokenizer to separate the parts by the dollar signs to get the dates and times separate. We then used another tokenizer to separate the month, year, and day out of the "from date" and "to date." The last tokenizer separates the "from time" and "to time" into hours, minutes, and seconds. The tokenizers store the values of the minimum and maximum process times in a series of six ArrayLists. ArrayLists are objects in Java that allow us to list a lot of values without knowing the length of the list, so it was a better choice over using an Array. In order to use an Array, it is necessary to know the size of the list from the beginning of the program. These ArrayLists are all Wrapper classes of integers, with the minimum seconds and maximum seconds being of type Double. Wrapper classes allow programmers to treat "primitive values" such as integers or floating-point numbers as objects, since ArrayLists have to contain objects and can't store the numbers by themselves. These values are checked for a "0." If a value such as "0 seconds" or "0 minutes" is present, the function changes them to an empty string so they do not appear in the final table the user sees. They are also all multiplied together and converted into seconds to make comparison with the user's range much easier. It would be a lot more difficult to individually check to see if the months, days, hours, and seconds were greater than the corresponding parts of the actual data. The conditional logic to do so would be too complicated to implement effectively. If all the data were displayed without the logic to remove them, there wouldn't be enough room horizontally for the tables to render at a font large enough to read.

Vertically formatting the tables so that the processes fit was a challenge as well. In the application, the tables are all scrollable and dynamic so each time the user changes the preferences, the table data reloads. This was made possible by using a method to clear all the rows of the table as soon as any of the preferences are changed. Without this, the values from the previous query would still be in the table along with the new information. In order to do this, XML was simplified with only a RelativeLayout nested inside a ScrollView. RelativeLayouts in Android allow users to make form elements appear on the screen according to where the other elements are. Telling the computer to display certain elements below or above others, or exactly 30 pixels away from another element are possible, which makes formatting a table easier. The ScrollView is a layout that simply allows the screen to be scrollable. The innermost layer is an empty table layout with the code generating Tablerows and Textviews as they are received from the server. Tablerows are layouts that are always nested in a table, and Textviews are basically just labels that text goes into for display purposes. If the user goes back to the pie chart screen or any of the tables, and there is a change in the date/time range or threshold for min/max or number of processes, the TableLayout is cleared, and the rows and texts are regenerated.



[11] A screenshot of the old data visualization method Telcordia was using before to view the start of a particular process

4 Results

The application allows a manager to quickly gain knowledge and take early actions to yield results. The largest accomplishment of the application is that it is able to fetch live data from a server and render it in a user friendly way. Data that was almost impossible to understand can now be displayed in an organized form. The data is displayed in many different forms, all of which should be familiar to the user. Displaying the data in different ways allows a manager to easily recognize what he or she is looking at based solely on screen design. At a quick glance, a user is able to find out where work should be focused and any business proceedings that may be holding up the company. It is important to provide a manager with information as quick as possible so that he or she can take actions and elicit changes. The code is designed in such a way that he or she can easily be repurposed by Ericsson and used to contact its actual servers. Mobile Exec serves as a foundation that Ericsson can build upon and expand to model new information to increase overall efficiency and productivity.

5 "Discussion/Analysis":

Not only is Mobile Exec about streamlining Ericsson's business, but also it is about developing a framework for future mobile applications. Data visualization is a core aspect of any company. Stats alone mean nothing until they are put in applicable, understandable forms. It is important to streamline data and tailor it to fit a specific audience. Thousands of hours of research and work are worthless if it is not presented in the correct manor. Data visualization ensures that all of the time spent gathering and compiling information will impact the future and lead to innovation. Mobile Exec shows how a large volume of data can be logically organized in an understandable and effective manner. At the core of this project, is an integration of design, efficient app design, and powerful network framework. This overlap creates a canvas for the future of business where powerful, simple mobile applications will provide information that revolutionizes how companies are managed.

6 Conclusion

When designing the application, the focus was placed on both aesthetics and how to make the application universally accessible and able to run effectively on numerous devices. A balance of these goals led to the creation of an application that is both effective and easy to use. Mobile Exec also served as an example of how large amounts of data can be fetched from a server and displayed on a mobile device. Ericsson will most likely build upon Mobile Exec and modify it to fulfill an assortment of tasks. Our application serves a prototype for the future data processing of corporate America.

7 Acknowledgements

We would like to thank our project advisors, Christine Hung of Ozmosis Learning as well as Byung-Woo Jun and Michael Davies of Ericsson for guiding us through our design. In addition, we would like to thank our RTA mentor, Amanda Rumsey, the Assistant Director of the Governor's School of Engineering and Technology, Jean Patrick Antoine, the Governor's School Program Coordinator Dean Ilene Rosen, research coordinator Stoyan Lazarov, and head RTA Adrien Perkins, and the NJ Governor's School

Board of Overseers. Finally we would like to thank the sponsors of the 2012 School of Engineering and Technology: Rutgers University, Rutgers University School of Engineering, State of New Jersey, Morgan Stanley, Lockheed Martin, Silver Line Building Products, South Jersey Industries, Inc. and PSE&G.

8 References

[1] "Philosophy and Goals." *Source.android.com*. N.p., n.d. Web. 12 July 2012.

[2] "Colorblind Home Page." *Testing Color Vision*. N.p., n.d. Web. 21 Jul 2012. <http://colorvisiontesting.com/>.

[3] Myer, Tom. "SitePoint Â» Learn CSS | HTML5 | JavaScript | Wordpress | Tutorials-Web Development | Reference | Books and More." *Sitepoint.com*. N.p., 24 Aug. 2005. Web. 12 July 2012.

[4] Fallon, Eugene. “What is BPM?.” AIIM. Keynote.

[5]Rouse, Margaret. "SOAP (Simple Object Access Protocol) ." *SearchSOA*. N.p., September 2005. Web. 21 Jul 2012.

[6]."ksoap2." *http://code.google.com/p/ksoap2-android/*. Google Project Hosting, n.d. Web. 21 Jul 2012. <http://code.google.com/p/ksoap2-android/people/list>.

[7]Jenny, Bernhard. *Color Oracle*. N.p., 8 December 2011. Web. 21 Jul 2012. <http://colororacle.org/>.

[8] "Dashboards | Android Developers." *Developer.android.com*. N.p., n.d. Web. 12 July 2012.

[9]Ort, Ed, and Bhakti Mehta. "Java Architecture for XML Binding (JAXB)." *Oracle*. Oracle, March 2003. Web. 21 Jul 2012. <http://www.oracle.com/technetwork/articles/javase/index-140168.html>.

[10] . "tcpmon." *Google Project Housing*. Apache, n.d. Web. 21 Jul 2012. <http://code.google.com/p/tcpmon/>.

[11]“Project Introduction 070512.” PowerPoint presentation. SERC, Piscataway, NJ. 6 July, 2012.