A Prototype Mobile Expert System for Nutritional Diagnosis

Christian Quesada, Marcelo Jenkins
Center for ICT Research, University of Costa Rica, San Pedro, Costa Rica
cristian.quesadalopez@ucr.ac.cr, marcelo.jenkins@ecci.ucr.ac.cr

Abstract

This paper describes NUTRITION UCR, a prototype expert system for human nutritional diagnosis developed in Java on Android using a service-oriented architecture. The system runs on mobile devices and offers smart features that evaluate the nutritional condition of an individual by assessing their physical characteristics and eating habits. We explain the knowledge engineering process used to develop the system, overview the system architecture and selected design tools, and summarize some preliminary results from the prototype implementation.

Keywords—mobile software engineering, expert systems, web services, nutritional diagnosis, obesity.

Introduction

NUTRITION UCR is a prototype expert system for diagnosing, controlling, and monitoring human nutrition. The system assesses the physical characteristics of the user to determine their nutritional status and makes recommendations for reaching nutritional requirements and a balanced diet, as a consequence generating a knowledge database with the nutritional status and dietary habits of a university population. The system generates challenges, alerts, and constantly motivates the user to use the application and improve their nutritional habits. The expert system is implemented using the JESS, Java Expert System Shell, libraries (Laboratories 2012) and the Java programming language running as a Web Service on a Linux Web Server. The prototype calculates the BMI, Body Mass Index, as in Eq. 1 (OMS 2012), the ideal weight and physical contexture, frame size (Rivas 1991) and uses dietary information from (Bermudez 2012). This is our base for nutritional diagnosis on the proposed prototype. This prototype was developed for administrative staff, educators, and students of the University of Costa Rica as a tool to improve their eating habits and nutritional wellbeing. The goal is to incorporate the use of this application into their daily lives and help them acquire and maintain healthier eating habits.

The problem

The eating habits of today's Costa Rican society are alarming. Obesity rates have increased making our country one of the most obese populations in the Latin American region. Six in ten people suffer a disproportionate increase in their weight because their poor nutritional habits (CACIA 2012). The prevalence of obesity, BMI > 30, is 59 percent (Rosero 2009). Studies estimates Costa Rica will be one of the ten most obese populations in the world by 2020 (Euromonitor 2011).

This reality makes it essential to raise public awareness about the need for a much needed dietary improvement and encourage preventive care. Many people, and particularly students, cannot afford to consult a private nutritional expert (Morales 2012). Hence, public health agencies such as the Office of Welfare and Health of the University of Costa Rica face the challenge of finding alternative methods for educating the population in incorporating healthy eating habits into their daily. One problem in healthcare is the lack of availability for frequently health monitoring. Health software offers less expensive solutions reducing the physician-patient physical relation and provides monitoring solutions. Mobile Internet and the use of Web for medicine have a strong impact on health-care models that are based on the concept of anytime and anywhere connections. Mobile software applications can help facilitate the distribution of nutritional information, learn to assess their own nutritional level, and acquire better eating habits to improve their current condition (CIGA 2007) and (Salud 2003).

Related Work

According to the Academy of Nutrition and Dietetics (Academy 2012) there are several applications for iPhone and Android devices that allow users to take care of their

diet and to control their nutritional goals according to the user ideal weight. For example, Calorie Counter, Calorie Counter & Diet Tracker, Diet Point and Daily Burn are applications that allow counting calories and contain food databases and allow monitoring of weight loss. The most important characteristic of Nutrition UCR project is that it was created as a specific tool for the population of the University of Costa Rica, which means that the information provided is really useful for the everyday university community user. Besides, the system generates a database of knowledge to the health authorities of the university that can be used for decision making of university policies in the area of health.

The System Development

We delineated the scope of the application working with two experts: Ana Yancy Zuniga and Sonia Vargas from the Health and Welfare Promotion Office from the University of Costa Rica. The design and development were carried out using the basic principles of agile methodologies (Scrum 2012). A prototype was built and initially evaluated with the nutritionist and two end users, one male (Christian Quesada) and one female (Elizabeth Sanchez) for a period of three months. The prototype is divided into three main components: (1) nutritional diagnosis, (2) healthy eating plan (cafeterias and menus) and (3) control and monitoring (nutrition education).

The Android client uses the expert system nutrition web services. The application also provides an update service through internet for the regular information displayed in the application. It is updated periodically through the services offered by a central computer that provides updates of the database of knowledge about nutrition and data of cafeterias and menus. The system communicates with the central server to update the patient record data and progress related with their healthy eating plan.

System Architecture

The prototype system architecture and others components related with whole system are presented in Figure 1.

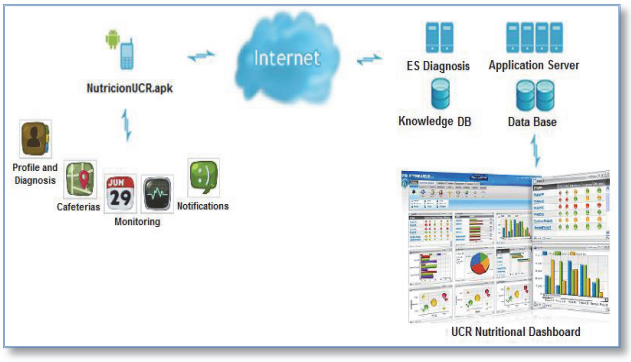

Figure 1. System Architecture.

It includes user profile, nutritional diagnosis, healthy eating plan (cafeterias and menus), user control and monitoring (nutrition education) and the university nutritional Dash Board.

Java and Expert Systems

Java allows deploying internet-based expert systems. JESS is a rule-based engine and scripting environment implemented in Java by Sandia National Laboratories (Laboratories 2012) and initially inspired by CLIPS (Giarratano 2007). Using JESS scripting language is possible to program expert systems in Java capable of reasoning using knowledge databases implemented as declarative rules. JESS supports the development and execution of forward-chain rules, which are compiled and executed using the RETE algorithm created by Dr. Charles L. Forgy (Rete 2012), a pattern recognition algorithm to deploy fact-based expert systems. JESS also allows access to the Java APIs to create and manipulate Java objects. For coding JESS knowledge rules in Java, the Java source code instantiates JESS classes to use the RETE algorithm. The knowledge database file is loaded to run the engine. The method Run() executes the RETE algorithm that returns the results. In this case, JESS engine is instantiated from Java class using RETE algorithm. The interaction between JAVA and JESS exchange objects instances for the facts and results. The Android application sends information about the user and his/her characteristics and the expert system evaluates and generates the diagnostic for the Person object. The expert system is accessible as a web service in JAVA and AXIS2 (Axis2 2012) and implemented on a Glassfish Linux Server. It receives and sends the JAVA objects in the JSON interchange format. The web service program also supports Windows servers IIS and Apache or Tomcat.

Nutritional Diagnosis

The nutritional diagnosis has two key elements; the expert system that provides its functionality as a web service on an external server, and the client application running on Android devices that interact with the end user.

The prototype diagnoses the patient as a result of an inference process that requires the following inputs: gender (Male, Female), age in years, weight in kilograms, height in meters, and wrist diameter in centimeters. The expert system returns the results to the user in natural language, which in turn is registered on the central server for review by the nutritionist. The prototype calculates the BMI as in Eq. 1. These calculations are coded as rules in the system knowledge database.

$$\text{BMI} = \left(\frac{\text{Mass(kg)}}{(\text{Height(m)})^2}\right) = \left(\frac{\text{Mass(lb)}}{(\text{Height(in)})^2}\right) \text{x } 703¤ \qquad (1)$$

Once the system analyzes the BMI result, it classifies the person according to interpretation of the body mass index. For example, if a user has BMI from 20 to 25 he/she has

healthy weight, but if he/she has BMI from 25 to 27.5 suffers overweight and probably feels fatigue, digestive diseases, heart problems, poor circulation in legs and varicose veins. Our system also calculates the ideal weight and the body frame size as in Eq. 2 and 3 according to (Rivas 1991). Table 1 and Table 2 below describe the interpretation of the body frame size and the math for ideal weight.

$$\text{Contexture Index} = \left(\frac{\text{Height(m)}}{\text{Wrist Circumference(cm)}}\right) \quad (2)$$

$$\text{Ideal weight(x)} = 48 + x\left(\frac{(\text{height} - 152)}{2.5}\right) \quad (3)$$

With the result of body frame size formula one person is categorized as a small, medium or large person. According to sex and body frame size classification it is calculated the ideal weight as in Table 2 and Eq. 3.

Sex	Formula	Body frame size
Male	>10.4	Small
	9.6-10.4	Medium
	<= 9.6	Large
Female	>10.9	Small
	9.9-10.9	Medium
	<=9.9	Large

Table 1 Contexture Index.

Sex	Body frame size	Formula
Male	Small	-10%
	Medium	(3) Ideal weight(x=2.7)
	Large	+10%
Female	Small	-10%
	Medium	(3) Ideal weight(x=2.3)
	Large	+10%

Table 2 Ideal Weight Formula.

The Expert System Server

The knowledge database is represented by a script in JESS rules and stored in a separate file from the application, allowing its continuous updating from the central server. As the knowledge database grows with new rules the system will be updated. The server returns the results of the diagnosis as a Java object that is represented in the exchange format JSON (Google-gson 2012).

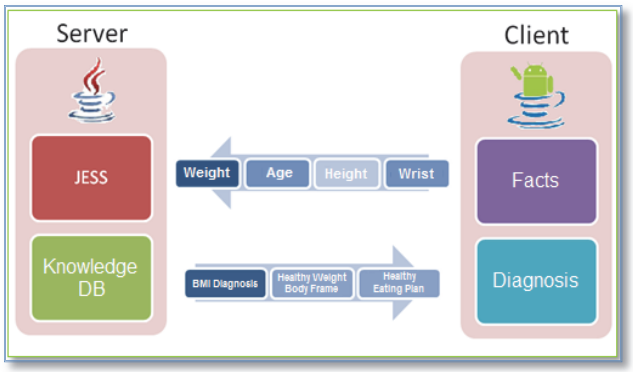

Figure 2. Expert System Model.

Figure 2 shows the model of the diagnosis expert system and its web service. The Android application sends information about the user and his/her characteristics and the expert system evaluates and generates the diagnosis for the Person object. This nutritional status and recommended diet plan is then returned as a result to the Android Application. The Person class has all the user-specific information and basic features for nutritional diagnosis. The Android application sends this object with the basic characteristics and the web service returns the diagnosis with the patient´s nutritional status in the same object.

Knowledge representation in JESS

The NUTRICION UCR expert system prototype contains almost sixty rules and ten functions to diagnose the basic nutritional status of a person, that is, BMI, Ideal Weight and Body Frame Context. For this prototype, the knowledge database was limited to this set of rules; these were coded in JESS because in the future the system might increase the number of rules to represent additional expert knowledge about nutrition. In consequence, updates could be made in the knowledge database without impacting the application´s code.

Due to space limitations, we only show a few sample rules in this paper. JESS scripts basically consist of facts, functions, rules, and queries. Facts are defined as lists which are asserted with assert or deffacts functions. The template for the fact must be defined before facts are used. Functions look like lists but are defined with deffunction. Users can define their own functions by means of other Jess functions or by writing them in Java. Rules consist of if-part and then-part. The '=>' stands between if-part and then-part. All the lists in the if-part are interpreted as facts and all the lists in the then-part are interpreted as function calls. If there are more than one fact in the if-part they are implicitly connected with logical AND. Queries are defined by means of defquery function and consist only of if-part. Queries return an iterator over the found facts. With the rules that describe the problem written in Jess, the rule engine will find the solution automatically.

The first step is to define data structures to represent the smallest pieces of a possible solution. Here it is needed to represent the Person and Diagnosis classes. A deftemplate is like a class declaration in Java, while class objects have member variables, deftemplate have slots. For instance, the person template has slots named sex, age, mass, height and wrist as is show in the following script:

```
(deftemplate result-bmi
  "Result BMI to diagnose Person nutrition status"
  (slot bmi-person
     (allowed-values
     Very-severely-underweight Severely-underweight Moderately-
  Underweight Underweight Normal-healthy-weight Overweight
  Moderately-obese-Class-I Severely-obese-Class-II Very-severely-
  obese-Class-III))
(slot bmi-msg (type STRING)))
```

The program in Java uses these templates to create facts for each of the possible combinations. For example:

```
private void DefPerson
     (String sex, double age, double mass, double  height, double wrist)
{
    try {
        Fact myFact = new Fact("person", engine);
        myFact.setSlotValue
              ("sex", new Value(sex, RU.SYMBOL));
        ("age", new Value(age, RU.FLOAT));
        ("mass ", new Value(mass, RU.FLOAT));
            ("height ", new Value(height, RU.FLOAT));
            ("wrist ", new Value(wrist, RU.FLOAT));
        engine.assertFact(myFact);       }
    catch (JessException e)  {
      result = "(7.1):" + e.toString();
       }}
```

Diagnosis knowledge can be represented fairly well using flowcharts and then transform these flowcharts to Jess rules. For example, Fig. 3 shows part of BMI flowchart for diagnostic process.

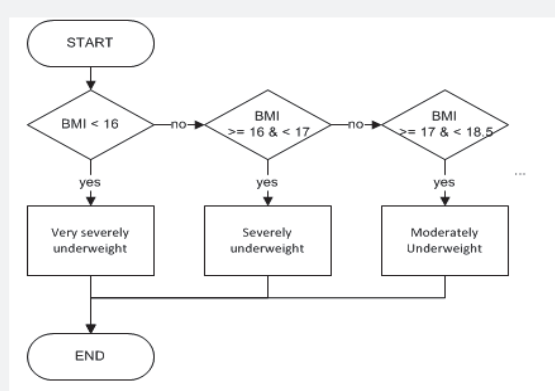

Figure 3. Flowchart representing part of BMI diagnosis process.

The Jess rules that capture the logic from Fig 3 are the following. The calc-BMI function described in (1) is used in all BMI diagnosis rules. It receives as parameters the weight and height and returns the body mass index that is the basis for diagnosing the nutritional status of a person. It also implements the rules bmi-diagnosis-#-# which indicates the result of the diagnosis.

```
(deffunction calc-BMI (?m ?h)
      (return (/ ?m (* ?h ?h))))
(defrule bmi-diagnosis-0-0
      (person (mass ?m)
      (height ?h&:(< (calc-BMI ?m ?h) 16.0)))
      =>(assert (result-imc
          (bmi-person Very-severely-underweight)
          (bmi-msg "Prostration, fatigue, weakness, and degenerative
      diseases threatening."))))
```

Figure 4 shows part of Ideal Weight and Body Frame Size flowchart for diagnostic process according to Table 1 and Table 2. The JESS rules that capture part of the logic from Fig 3 and Fig. 4 are as follows:

```
(defrule body-frame-size-diagnosis-0-0
    (person (sex Male) (wrist ?m)
        (height ?a&:(>= (/ (* ?a 100.0) ?m) 10.4)))
    =>(assert (result-body-frame
        (body-frame-person Small)
        (body-frame-msg "Small Body Frame."))))
(deffunction ideal-weight-male-medium (?a)
    (return (+48.0(*(/(-(* ?a 100.0)152.0)2.5)2.7))))
 (defrule ideal-weight-diagnosis-0-0
    (result-body-frame (body-frame-person Medium))
    (person (sex Male) (height ?a))
    =>(assert (result-Ideal-Weight
        (min-weight (ideal-weight-male-medium ?a))
        (max-weight (ideal-weight-male-medium ?a)))))
```

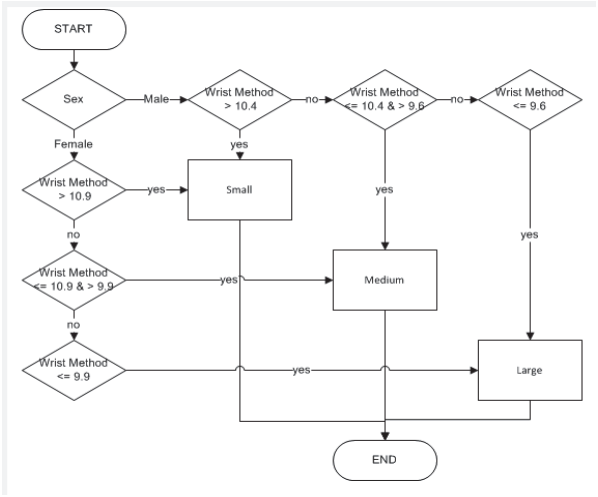

Figure 4. Flowchart representing Body Frame Size diagnosis process.

The Prototype Application

We explain here only a small portion of the implemented functionality. A sample video of the prototype´s functionality can be seen at http://www.youtube.com/watch?v=8GyvxsDeFT8. The NUTRITION UCR prototype includes user profile, nutritional diagnosis, healthy eating plan (cafeterias and menus) and user control and monitoring (nutrition education) as in Figure 5. This application runs in Android devices.

Nutritional Diagnosis and User Profile

Figure 6 (a) shows the result of the nutritional evaluation and Fig. 6 (b) shows the screen with the recommendation of diet according to nutritional status of the user; this gives the detail recommended calorie intake for each meal times.

Healthy Eating Plan (Cafeterias and Menus)

The prototype offers a list of places to eat on campus (cafeterias). The system offers healthy meals according to

the recommended diet to the user. For each of the locations the application allows make a phone call, go to the website of the cafeteria, and send an email. The system allows interaction through Google maps; the system shows the location of the cafeteria in the campus and the route to go to the cafeteria. Finally, the system lets the user view a general map of locations in the university campus. Once a university cafeteria is selected, the user can explore the menu offered and for each of the meals the system gives a detailed nutritional description.

Figure 5. Android Application

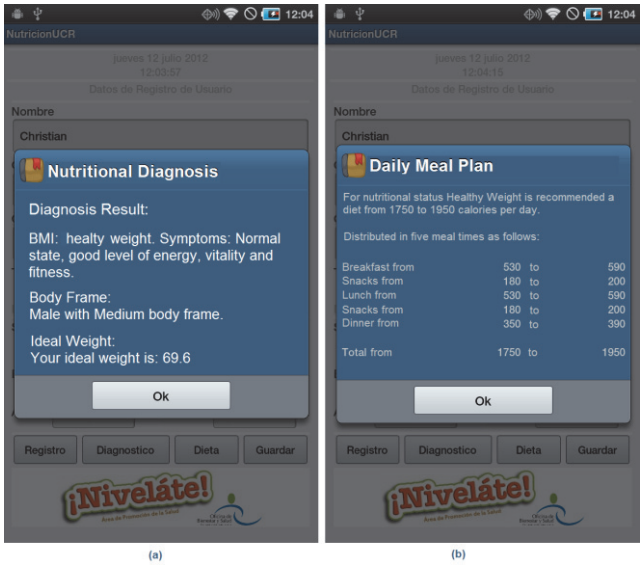

Figure 6. (a) Nutrition Diagnosis (b) Recommended Diet.

Control and monitoring (Nutritional Education)

The application notifies the meal time that the user must perform and then allows him/her to search for healthy food places. With the details of the selected food, the user can use the location options. Also notify the necessity to drink water and work out.

Figure 7. Calendar with nutritional results.

As shown in Fig. 7, the system record daily events with the following user information: five healthy meal times a day (breakfast, morning snack, lunch, afternoon snack and dinner), one or two times of physical activity (thirty minutes each), eight glasses of fluid a day, weight for day and sexual activity log. Depending on the activities performed by the user throughout the day, the system calculates its nutritional performance indicator (*Nivelómetro*). This evaluates your daily activities and score them using a five-color code. Depending on the activities performed by the user throughout the day, the system calculates its classification. The system lets the user check all historical information performed.

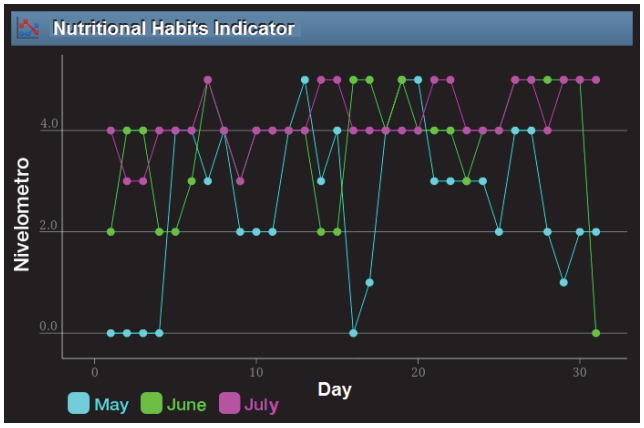

Figure 8. Statistics for Nutritional Indicator.

The system provides a comparison of three months showing a comparative graph, as shown in Figure 8. It shows the performance of the user for each month (i.e. May, June and July) with the results of the indicator that measure the performance of the user in terms of nutritional habits. Figure 8 shows that while there was some habit inconsistencies throughout the first month, the trend improved over the next two months. The prototype uses the package (Chartdroid 2012) for represent the data in a plot graph.

Conclusions and Future Work

The system is a useful tool for educating users on nutritional related topics and could potentially allow the creation of the campus nutritional status database of approximately 39,000 college students. The end-user application was built on an Android platform with a simple and efficient human-device interface. Content updating methods were written using HTTP and SOAP. The prototype was built using only open source tools and services on the Android platform: geo location, web query, text messaging, phone call, email notifications, and the operating system facilities.

The application validation was performed through exhaustive running experiments. In the testing phase, we considered the prototype functionality and user experience. Real Android devices were used in these tests. Early prototype evaluations were conducted on two non-nutritional-expert users. The end users or test subjects showed positive results when using the application for a period of three months (increasing the nutritional performance indicator). The system caused an improvement on the eating habits of the two users by focusing their constant attention on the healthy activities recommended by the program during a three-month period. While there were some habit inconsistencies throughout the first month, the trend improved over the next two months (see figure 8). The incorporation of an automatic notification feature caused a significant improvement in the third month.

Despite the encouraging results, additional experimentation and a larger sample are needed. The expert system must be published on the university servers, and then the prototype will be evaluated using a more representative sample of the university population. It is also important to measure the number of users that use the Android platform, iOS, or Windows Phone platforms to analyze the feasibility of offering the system to users of other mobile devices. Another point to be considered is the expansion of the knowledge database by including more specialized nutritional information and the validation of information and parameters by additional nutrition experts. Last, it would be interesting to incorporate some machine learning capabilities (i.e. neural networks). The system could learn from each user and use this information to enhance future diagnoses and diets.

Acknowledgment

This research was also supported by the Computer Science Department at the University of Costa Rica.

References

Laboratories, S. N. (2012). JESS, the Rule Engine for the JavaTM Platform. Downloaded January 1st, 2012, http://www.jessrules.com/jess/index.shtml

OMS. (2012). Obesidad y sobrepeso. Downloaded January 1st, 2012, http://www.who.int/mediacentre/factsheets/fs311/es/

Rivas, H. N. (1991). Manual de Dietas para uso en la Consulta Nutricional de la Oficina de Salud de la Universidad de Costa Rica. San Jose.

Bermúdez, A. Y. (2012). Campana Nivélate. Downloaded February 1st, 2012, http://www.sais.ucr.ac.cr/mmmsaludable/index.html

CACIA. (2012). XII Congreso Salud y Nutrición. Cámara Costarricense de la Industria Alimentaria.

Rosero Bixby, Luis G. B. (2009). Obesidad, Envejecimiento y Mortalidad. San Jose: Centro Centroamericano de Poblacion.

Euromonitor Internacional. (2011). Consumer Lifestyles in Costa Rica, Obese and Overweight Population. Euromonitor Internacional.

Morales, D. A. (20 de 2 de 2012). Consulta de Nutricion. (E. S. Jiron, Interwiwer)

CIGA, C. I. (2007). Actualización de lineamientos técnicos para la elaboración de Guías Alimentarias. San Jose: Comisión Intersectorial de Guías Alimentarias para Costa Rica.

Salud, M. d. (2003). Encuesta basal de factores de riesgo para enfermedades no transmisibles. Módulo 1: Factores Alimentario Nutricionales. Obtenido de Ministerio de Salud de Costa Rica: http://www.ministeriodesalud.go.cr/index.php/boletines/doc_view/390-encuesta-basal-carmen-cartago-factores-alimentario-nutricionales-cr?tmpl=component&format=raw

Academy of Nutrution and Dietetics, Media Press Room Read. (2012). Recuperado el 20 de 07 de 2012, de Academia de Nutricion y Dietetica: http://www.eatright.org/Media/content.aspx?id=6442467041

Giarratano, J. C. (2007). CLIPS User Guide. Recuperado el 01 de 04 de 2012, de Clipsrules Sourceforge: http://clipsrules.sourceforge.net/documentation/v630/ug.pdf

Rete. (2012). Rete Algorithm. Downloaded on May 3rd, 2012: http://www.jbug.jp/trans/jboss-rules3.0.2/ja/html/ch01s04.html

Scrum. (2012). Downloaded May 1st, 2012: http://www.proyectosagiles.org/que-es-scrum

Google-gson. (2012). Downloaded March 3rd,2012, http://code.google.com/p/google-gson/

Axis2, Apache Java. (2012). Downloaded March 3rd,2012, http://axis.apache.org/axis2/java/core/

Chartdroid, Google Code. (2012). Downloaded June 4th, 2012, http://code.google.com/p/chartdroid/