The Report committee for Solomon Mugume Rugunda

Certifies that this is the approved version of the following report

Mobile Application Development in Africa

APPROVED BY

SUPERVISING COMMITTEE:

Supervisor: _____
Suzanne Barber

_____
Thomas Graser

Mobile Application Development in Africa

by

Solomon Mugume Rugunda, B.S EE

Report

Presented to the Faculty of the Graduate School

of the University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

December 2010

Abstract

Mobile Application Development in Africa

by

Solomon Mugume Rugunda, MSE

The University of Texas at Austin, 2010

SUPERVISOR: Suzanne Barber

This work takes a look at how some of the challenges facing the African Continent can be tackled by the use of Mobile Phone Applications. Mobile phone penetration in Africa is very high, as indicated by statistics from South Africa in 2009 which showed that 70% of all South Africans owned a cell-phone. As such, the mobile phone is the ideal infrastructure platform to introduce technology applications that can be easily accessed by the general public. The paper briefly discusses the current state of mobile applications on the continent including some key applications presently deployed and how they are being used to meet some major social challenges faced. The work then proposes the creation of the Mobile Doctor Application, a tool that gives simple solutions to end-users in response to their health related text message queries. The key stakeholders for this application and their respective roles are then discussed. The paper then details the requirements for the application, separates them logically and represents them graphically for a clear understanding. Also put forth are two possible different sets of architecture for this application, one fully manual and the other using different sets of technologies. Finally, each architecture is evaluated using compliance metrics and the ATAM Quality Attribute Tree

Table of Contents

List of Tables

List of Figures

I. INTRODUCTION

Today in Africa, majority of the people live in rural areas. One of the main problems they face is a basic lack of important and on-time health information both regarding people's health and their animal's health, considering that many people are cattle farmers. When a child gets sick, sometimes parents don't know how to respond in the medically correct way, e.g., a child develops a high fever and the unsuspecting mother will put a blanket over the child to try to help them feel better, but clearly this only raises the child's temperature and makes the illness worse, thus illustrating the problem of a lack of correct health information on how to remedy diseases or symptoms of illness. Similar problems are illustrated with the cattle that farmers raise. The farmer may notice that one particular cow is not producing milk anymore and showing symptoms of sickness. In many cases, the farmer doesn't really know what to do until the veterinary doctor arrives to care for the animal.

There is also the problem of transportation in rural areas. In many cases, the doctor lives far away from the patient. The problem is exacerbated by the poor roads that make transportation very difficult and as a result, the time taken for a patient to be transported to a doctor takes a lot longer than it should. During this time a patient's condition can significantly deteriorate if nothing is done to help the patient.

Going hand in hand with this lack of information problem is a lack of money to handle expensive health costs. Many of the locals make very little money and cannot afford a doctor's high consultation fees. To make matters worse, a doctor in many cases doesn't do much more than give them very basic home-remedy solutions that they could have applied a lot earlier if this information had been available.

To address these problems, this paper proposes the creation a Mobile Doctor application that will provide affordable, on-time remedy and first aid health information to patients and to farmers whose animals are sick. The relevant requirements, architecture and implementation details for the

Mobile Doctor Application are also discussed. This application will potentially save lives by providing easily accessible first response health information that one can administer themselves while they haven't yet gotten access to a doctor. The application will be similar to a text messaging application version of the popular WebMD website. Mobile Doctor will also save people money because the service will be a lot more affordable than the doctor's consultation fees. The application is not meant to replace a doctor, but instead provide home remedy and first aid solutions while people are not able to reach the doctor.

The idea behind the application is that the farmer should be able to use their cell-phone to send a simple text message or SMS (Short Message Service) query to the Mobile Doctor number, indicating in a few words the cow's symptoms or suspected disease. He will then receive a reply on how to respond correctly to those symptoms. Similarly, the mother whose child is sick enters "High Fever" as text into the phone and sends this message to the Mobile Doctor number which will return the home remedy or first aid ways of how to respond. The user will be able to enter a disease name, a symptom name or a phrase regarding what the problem is. The system will be intelligent enough to discern key words and know exactly what the customer is trying to find out.

To come up with a database of medically accurate responses that would be sent to end-users of the application, a team of highly experienced doctors would be used to provide medical solutions to common health problems. As the latest medical information changes and new discoveries are made, the database will be updated so that end-users have the latest information at their fingertips.

On top of sending responses to customer queries, the application will have the ability to "push" daily health tips to end-users. This could be a paid-for service or a free service to customers that already pay a monthly fee for query responses. An alternative way of charging customers could be per text message received.

The Mobile Doctor application will in effect solve the following major problems:

1. Save time in the context of health – Taking the case of the aforementioned case of the mother with a child who has a high fever, if they decided to take the child to the doctor who is miles away, it may be too late by the time the child reaches the doctor. In contrast, imagine if the mother was able to simply send a text message with the child's symptoms to the Mobile Doctor number as soon as she sees the child is sick. She would get an almost immediate response with how to treat high fever, i.e., Drink cold liquids, place cool cloth over forehead, etc. The application would also return a similarly useful list of things not to do.
2. Save money – Doctors' high consultation fees that are not affordable to locals will in some cases be mitigated or avoided where the answers from the application solve the health problems being faced.

The rest of the report is organized as follows: Section 2 contains the Background and Motivation, Section 3 lays out the Stakeholders, Section 4 describes the Requirements, Section 5 details the Architecture, Section 6 provides the Results and Analysis and Section 7 presents the Conclusion.

II. BACKGROUND AND MOTIVATION

To get a sense of what is currently happening with mobile phone technology on the continent and appreciate the need for an application such as the Mobile Doctor Application, this section takes a look at the mobile expansion that has happened over the last several years, the subsequent rise of mobile application development and the reasons for having a Mobile Doctor Application

a. Mobile Phone Penetration in Africa

In the United States and in Europe, there has always been planned landline telephony infrastructure laid out. This has involved telephone wires being connected along roads, into communities and eventually into houses. There has also been a gradual overlapping progression from landline to mobile telephony and consequently, it is common to find many people owning both a mobile phone and having a landline at home. In Africa however, it has been quite different. Initially there were only a small percentage of people that owned landlines. It has been interesting to note that those numbers have barely risen as the years and subsequent economic development levels have increased. According to the International Telecommunications Union (ITU), there were fewer than 3 landlines per 100 people in 2009 in sub-Saharan Africa [2]. When mobile phones were introduced into the African market over the last few years, they caught on so fast that the growth of the landline "phase" was effectively skipped. The primary reason for this was the low cost of cell-phones and related infrastructure consisting mainly of broadcasting base stations as compared to the heavy costs incurred with installation of traditional landline telephony infrastructure and wiring all the way into homes [2]. As such the mobile phone market took off at an exponential rate in most African countries.

In South Africa in 2009, research showed that 70% of all South Africans owned a cell-phone. Consider this in light with some of the following figures during the same year [1]:

- 1.7% of South Africans had home loans (i.e., bank mortgages)

- 16% had a home telephone or land line
- 55% of South Africans had a bank account

These numbers show that despite the fact that the country is economically underdeveloped, the mobile phone penetration of 70% is disproportionately high. Also interesting to consider is the fact that the number of people of people with cell-phones in urban areas was 77%, which indicates that popularity of phones in rural areas, while not as high as the urban areas, was also fairly high. This illustrates the point that even the poorest carry these mobile devices. This is also demonstrated by the statistics of the same research group that show that 61% of the 70% of the owned phones are prepaid phones. While in America, most have contract phones, the opposite is true In Africa. It is the wealthy that carry the contract phones while the common people carry prepaid phones, which are not billed when a call is received but only when a call is made. Consequently, this high percentage of prepaid phones shows that phones are not just for the wealthy. It is not just South Africa, with over 40million users, exhibiting such numbers, but most of Africa with the key countries being Nigeria, Egypt, Algeria and others all having over 30 million mobile phone subscribers according to 2007 statistics [3].

Given such high numbers of mobile phone penetration on the continent, one might wonder whether there are other technology platforms that have similarly high dissemination. Internet enabled computers would be the other player in this arena, but statistics depicted in Figure 1 below show that the internet still has a long way to go to catch up with mobile telephones as the technology platform of Africa. People in developing nations don't have personal computers in their homes in the same way that those in developed nations do. This is primarily because of affordability. Figure 2 compares the relative monthly cost of a cell phone versus having broadband internet.

Judging from these numbers, it is clear that the mobile phone is the platform of choice when introducing new technology to African countries based on its high pervasiveness and low cost. This gives

credence to the potential success of the Mobile Doctor application and shows that the local people will have access to this technology through their mobile phones.

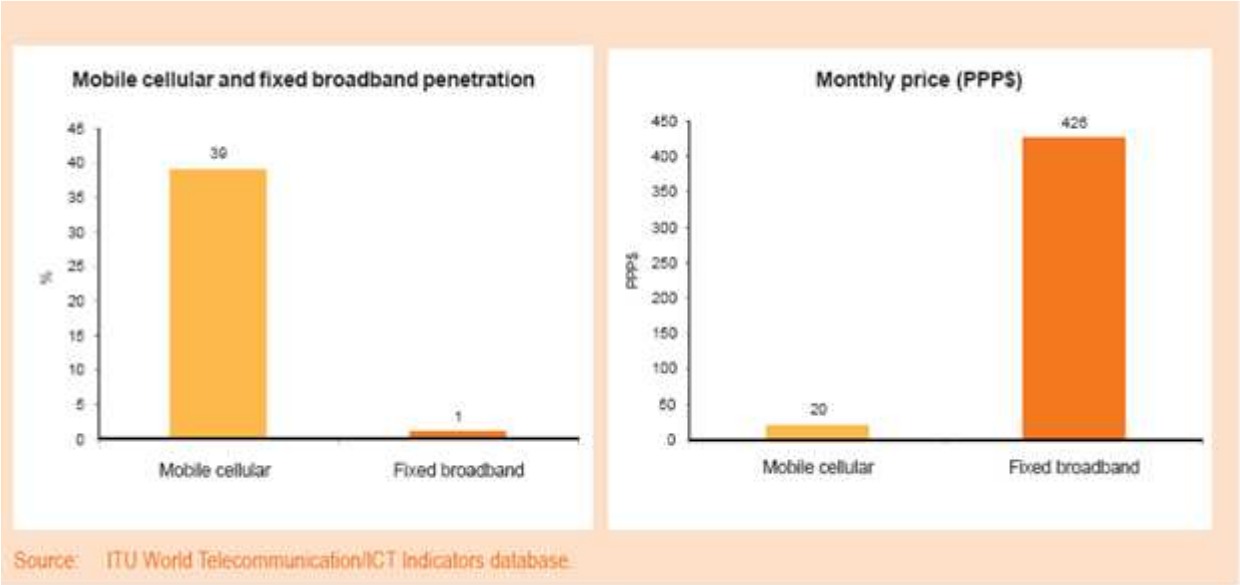

Figure 1: Mobile cellular and fixed broadband [1]

Figure 2: Monthly price of Mobile Cellular Vs Fixed Broadband in Africa [1]

b. Current State and Impact of Mobile Application Development in Africa

Given the huge popularity of mobile phones in the cities and villages, it is clear that this is the optimal infrastructure platform to introduce technology applications that need to be easily accessed by the general public. Over the past few years, young African software entrepreneurs have began to leverage the pervasiveness of these devices and have started developing simple applications that meet some of the basic needs of the population. There have also been American-based groups that have realized the potential of the industry and have taken the initiative to train young Africans in Mobile Application Development to benefit themselves and their continent. An example of such a group is the African Information Technology Initiative at Massachusetts Institute of Technology (MIT), which is a team primarily made up of MIT students that spend time in Africa teaching fundamental development

skills to young enthusiastic minds [4]. Some of the applications already developed have had a lot of success, indicating that there is room for more creative ideas of applications that will have a socially beneficial impact on the population. There are several examples of such existing applications. One example in Kenya [5] is an application created by Amos Gichamba, a Kenyan technologist, that helps farmers determine the daily prices of milk so that they are not cheated. Farmers, who live far away from the market places, use middlemen to transport their milk to market or dairy corporations. A typical situation would involve a day when a liter of milk costs 50 shillings at market price. The middle man will tell the unsuspecting farmer that the price is in fact 25 shillings and thus buys it from the farmer for a very low price only to resell it at a high profit to the end buyer. The application created by Gichamba allows a farmer to send a text message query to a database inquiring the daily price of the milk and by doing so, farmers avoid being cheated.

Another application that demonstrates the potential for success of mobile applications in Africa is M-Pesa in Kenya. “Pesa” means money in the local language of Swahili. M-Pesa is essentially a phone banking application that allows locals to send, receive and deposit money as well as pay bills over the phone. This is especially useful because it is not uncommon to find locals who are not able to access banking services from banks for several reasons. In some cases, banks are just not accessible in rural areas with poor transportation. In other cases, people don’t have the basic documents required to open a bank account. These people however have the same needs like everybody else. They need to store their money, send money to their children elsewhere or receive money from other relatives. M-Pesa steps in and creates an avenue for these financial transactions to be performed. As an example, one desiring to send money over their phone simply goes to the M-Pesa menu and selects “Send Money” and enters the recipient’s phone number, the amount to be sent and a secret PIN. Once this information is entered, the money is sent to the recipient and a confirmation text message is delivered to both the sender and the receiver. The receiver then goes to a local registered outlet to receive their

money. Most retailers can register as outlets, i.e., grocery stores, department stores, etc.

Such applications as the two mentioned above demonstrate the impact and potential of mobile application development to better the lives of the population. The application presented in this paper, Mobile Doctor, will similarly be available to the masses through the same channels and be able to have an impact on the health and wellbeing of the locals. The application will enable patients to have on-time firsthand access to important health tips on how to administer first aid or home-care to themselves or their animals. This will be done through text message queries to a database in a similar way to the previously mentioned M-Pesa application.

c. Health Challenges and the Need for the Mobile Doctor Application

In rural areas in Africa, there is a basic lack of education. In many cases, the cultures are largely superstitious in the way they handle signs of illness and often have no exposure to basic health information. A clear example is the still prevailing belief in some parts of the continent that sexual intercourse with a virgin will cure someone who is HIV positive. That is clearly medically incorrect but it is not uncommon to find newspaper reports of HIV positive men having raped young girls in an effort to cure themselves of the disease. These same men most likely own mobile phones and if they had taken just a minute to send a text message to Mobile Doctor number saying “Will a virgin cure me of HIV?” they could easily have received a response back to their phones indicating that this was in fact false and a common misconception. In addition, the Mobile Doctor application will have the ability to tell users “What not to do” in addition to the “What to do” feature about a specific condition.

Worldwide, nearly 11 million children die every day from preventable diseases. Five countries alone make up half of that figure. Of those five, three are African countries namely Nigeria, Ethiopia and the Democratic Republic of Congo [6]. The pie chart in Figure 3 below from World Health Organization (WHO) below shows the leading causes of death in developing nations.

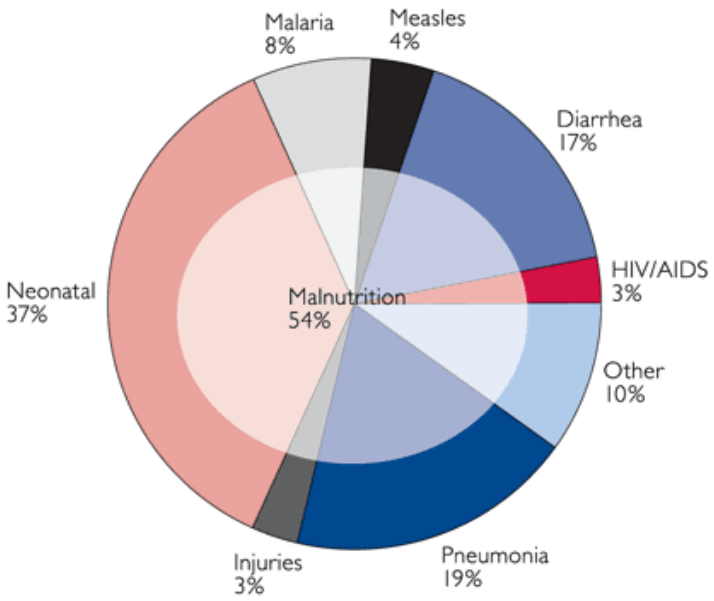

Figure 3: Leading causes of death in developing nations [7]

Among the main killer diseases are malaria, pneumonia, diarrhea and measles. These are diseases that can be prevented or otherwise managed fairly easily and can be largely mitigated with access to on-time basic health tips or home remedies based on the particular ailment. For diarrhea to be a leading cause of death is appalling and emphasizes the lack of health education. A simple application like the Mobile Doctor would easily tell a mother, upon reporting “Diarrhea” in her query, that the ways to treat this are to take frequent small sips of water to rehydrate the body, avoid certain oily foods that will make the stomach run, try yoghurt or other easy to digest foods, avoid dairy etc. Alternatively, an SMS query with the keyword “malaria” would return some home remedies for the disease that can be applied before a patient has access to a doctor. The application would also be able match up the disease with a list of common symptoms associated with malaria and how to handle them. For example, malaria is commonly associated with high fever. In this case, a response to the malaria query from the application could say something like “If fever present, soak cloth in cold water and place on head”. A child dying from preventable diseases is something that our Mobile Doctor application will be able to help with.

Unintended pregnancies are another major problem that crop up in rural areas. Locals are not educated on the use contraceptives and how to avoid unwanted pregnancies. On top of all this, in some cases, unsafe abortions are performed. Questions about pregnancy can always be sent to the Mobile Doctor which will return helpful answers.

All of the above are preventable diseases and conditions that can in some way be addressed by the Mobile Doctor application which will answer some of the patient's basic questions and give advice on how to tackle some problems that cannot be attended to immediately by a doctor.

Having presented the background and motivation behind the Mobile Doctor Application, the sections that follow delve into the technical details of stakeholder roles, requirements and architecture for the application.

III. STAKEHOLDERS

We can categorize the requirements for the Mobile doctor application based on what functionality is important to each stakeholder. To be able to do this, we need to delineate who the stakeholders are and what contribution they make to the overall system. The major stakeholders and their role with respect to the domain are as follows:

Customer

- This is the end user of the application. They are the typical person owning a mobile phone that enters a text message query trying to find the solution to their particular issue they are having.

Group of Doctors

- A group of certified and experienced doctors are used to build the database of responses to queries. They provide the expertise and knowledge that is used in the answers to the customer queries.
- When a new medical discovery happens or there is new medical information on a subject, the group of doctors is responsible for passing along these updates to the database entry personnel.

Project Manager

- Bears overall responsibility for the Mobile Doctor Application project delivery.
- Ensures the project is completed according to the budget.
- Ensures the project is completed on schedule.
- Works with the developers to set milestones.
- Gives regular reports to management about the progress of the project.

System Administrator

- Manages the day to day maintenance of the application and servers.

- Ensures that responses to queries are actually being sent and that any failures in the overall system are reported and dealt with.

Database Administrator

- Manages the database and ensures queries made to the database are being responded to.

Data-entry Employees

- Take the input from the group of doctors and enter the data into the system.

Software Architect

- Designs the overall design/structure of the system. The design should satisfy the non-functional requirements of the system.
- Makes the decisions on how the larger software system will decompose into smaller more manageable modules and how those parts will interact in a chosen environment.
- Communicates the architectural vision of the system to the developers.

Developers

- Write the code based on the architecture documentation.
- Write the unit tests and test cases for testers to implement.
- Performs maintenance on the code base based on changing requirements.

Testers / QA

- Identify and report bugs found during testing back to the developers.
- Perform unit and system tests.
- Ensures that the final product is error free.

Requirements Engineer

- Holds meetings with different stakeholders to assess needs.
- Builds the list of system requirements based on all stakeholder input.
- Organizes and represents requirements in documents, graphs, diagrams, etc.
- Gathers changes in requirements and passes them on to developers for integration.

Business Analysts

- Ensure there is business value to delivering the system, i.e., profitability.
- Set the pricing for the use of the Mobile Doctor Application based on market conditions and charges for related applications.

Data Analysts

- Parse and filter incoming SMS messages for key information.

Sales and Marketing

- Disseminate information about the application so that the public knows about it, is aware of the system capabilities and utilizes it.
- Create billboards, flyers and marketing campaigns for the Mobile Doctor Application.

Investor

- Supplies the needed capital to fund the project.
- Receives regular updates about the progress of the project from corporate executive.

Corporate Executive

- Represents the project at a corporate level.

- Receives reports from Project Manager about where the project
- Communicates progress of the project to the investor.

Legal Team

- Ensures that all the necessary medical liability information and risks are made known to the end-user of the application before they use the Mobile Doctor.

IV. REQUIREMENTS

The requirements of the system can be grouped into four categories [11]:

a- Functional Requirements – These represent the features that the system provides to the user. It is the set of functionality that the system must deliver to the end user.

b- Non-functional Requirements – These represent the non-tangible attributes that the system possesses, e.g., performance, modularity, reusability etc. It is the specific criteria that can be used to evaluate or judge the overall system.

c- Data Requirements – This refers to all the data that the Mobile Doctor application will need for its daily functions, i.e., Input to the application and output from the application.

d- Installation Requirements – This is refers to the type of infrastructure, platform and systems used to "house" the application, i.e., operating system, web / desktop environment, type of database.

e- Timing Requirements – This refers to the acceptable amount of timing that a particular feature of the application must take to execute. It could also be used to refer to the frequency with which a particular functionality happens as well as the pre and post conditions regarding a particular task.

a. Functional Requirements of the system grouped by stakeholder

Customer / End user – According the customer, the system should have these features:

1. Ability to respond with remedy information pertaining to symptoms entered for humans

   Data Input: Human symptom name, User input phrase
   Data Output: Remedy

2. Ability to respond with remedy information pertaining to symptoms entered for animals

   Data Input: Animal symptom name, User input phrase
   Data Output: Remedy

3. Ability to respond with correct remedy information pertaining to Human disease name entered

Data Input: Human disease name, User input phrase
Data Output: Remedy

4. Ability to respond with correct remedy information pertaining to Animal disease name entered

   Data Input: Animal disease name, User input phrase
   Data Output: Remedy

5. Ability to respond with “what not to do” regarding symptoms.

   Data Input: Symptoms, User input phrase
   Data Output: List of things not to do pertaining to symptoms

6. Ability to respond with “what not do” regarding disease

   Data Input: Disease, User input phrase
   Data Output: List of things not to do pertaining to disease

7. Ability to map user input symptoms and diseases text messages to appropriate section in Database

   i.e., An SMS pertaining diarrhea should not return remedy information pertaining to a headache.

   Data input: Symptoms/Disease
   Data Output: None

8. Ability to map symptoms entered to possible disease and vice versa.

   Data input: Symptom name, Disease name
   Data Output: Mapping

9. Ability to detect spelling errors entered by end user and respond intelligently.

   Data Input: User input phrase, Symptom name, Disease name
   Data Output: None

10. Ability to adapt to language. Because there are different languages in Africa, the system should be

    able to work with a few major different languages.

    Data Input: User input phrase, Symptom name, Disease name
    Data Output: None

11. Ability to display prevention information as an option to the user.

    Data Input: Disease name, Symptom name, User input phrase
    Data Output: Prevention Information

12. Ability to parse verbose user input phrase for key words and intelligently respond

Data Input: User input phrase
Data output: Keywords

13. Ability to push daily health tips to all registered phones for free.

    Data input: None
    Data output: Daily health tip

Group of Doctors - – According the doctors, the system should have these features:

1. System should present medically accurate information

   Data Input / Output: Latest medical information

2. Database should be regularly updated to display the latest information if new medical discoveries

   are found.

   Data Input: Latest Medical Information
   Data Output: None

Business Analyst – According the Business Analyst, the system should have these features:

1. Ability to charge the customer appropriately for the use of the application

   Data Input: Cost per SMS
   Data Output: Recorded Invoice

Software Architect / Developer – According to them, the system should have these features

1. Ensure that the characters responded by server fit into one text and that any SMS response that

   goes past 160 characters is broken at the 160the character by the program and transmitted and

   transmitted as a second SMS.

   Data input: Remedy, Prevention Information and Daily Health Tips
   Data output: Formatted Remedy, Prevention Information and Daily Health Tips

Legal Team – According the Legal Team, the system should have these features:

1. The system should present legal disclaimers every time a text message is received.

   Data input: None
   Data output: Legal Disclaimer

Database Administrator – According the Database Administrator, the system should have these features:

1. Ability to restrict/grant editing of database permissions.

   Data Input: List of authorized database editors
   Data output: None

2. Database entries can be created, edited and deleted.

   Data Input: Latest Medical Information
   Data Output: Database entries

GRAPHICAL REPRESENTATION OF REQUIREMENTS

Textual requirements can be represented graphically using several different models. Three models have been chosen to depict the above requirements namely the Operational Reference Model, the Task Decomposition Diagram and the Use Case Diagram.

Operational Reference Model (ORM)

The Operational Reference Model [11] is the highest level depiction of functionality showing the triggering event, the main high level system tasks and the terminating event. It is from these high level tasks that subsequent graphical models like the Task Decomposition Diagram derive smaller component tasks. Figure 4 below shows the Operational Reference Model.

Operational Reference Model (ORM)

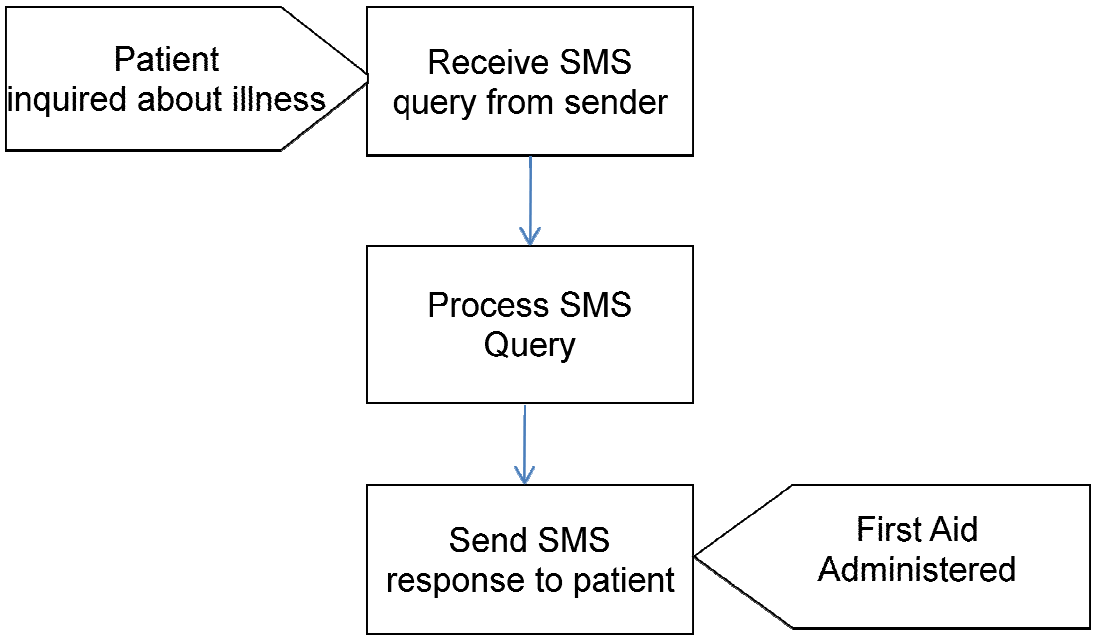

Figure 4: Operational Reference Model (ORM)

Task Decomposition Diagram

The Task Decomposition Diagram is a graphical model showing the breakdown of high level system tasks into smaller, more manageable components. This is very helpful when trying to identify individual units of work that need to be accomplished as they will be displayed in this drawing. These units of work typically drive scheduling. The task decomposition diagram is displayed in Figure 5 below.

Task Decomposition Diagram

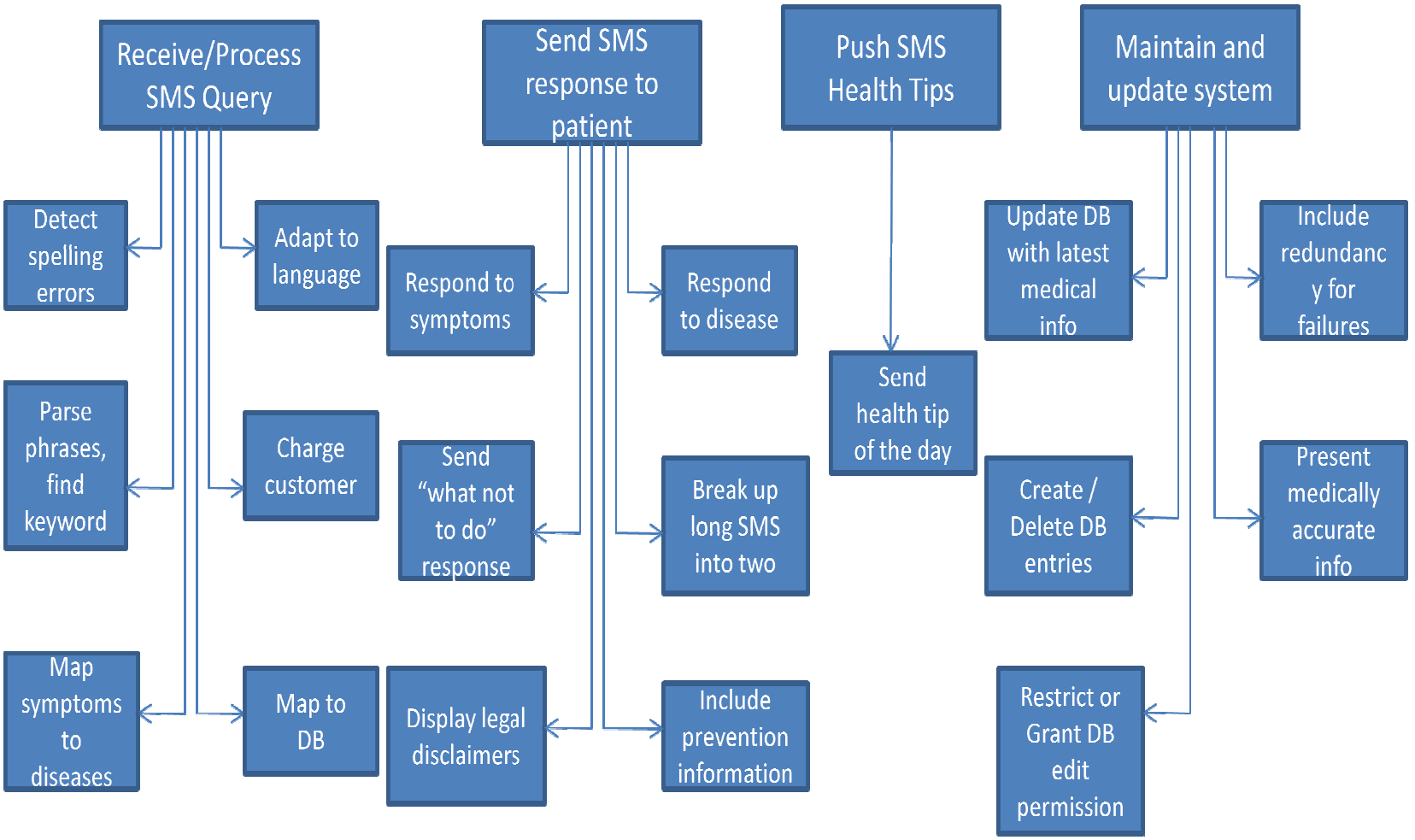

Figure 5: Task Decomposition Diagram

Use Case Diagram

A use case diagram is a graphical requirements model representing a typical usage scenario of an application. It also shows how actors or people involved with the system interact with it. Overall, the Use Case Diagram depicts a sequence of actions of typical usage of the application and how the actors are involved with these actions.

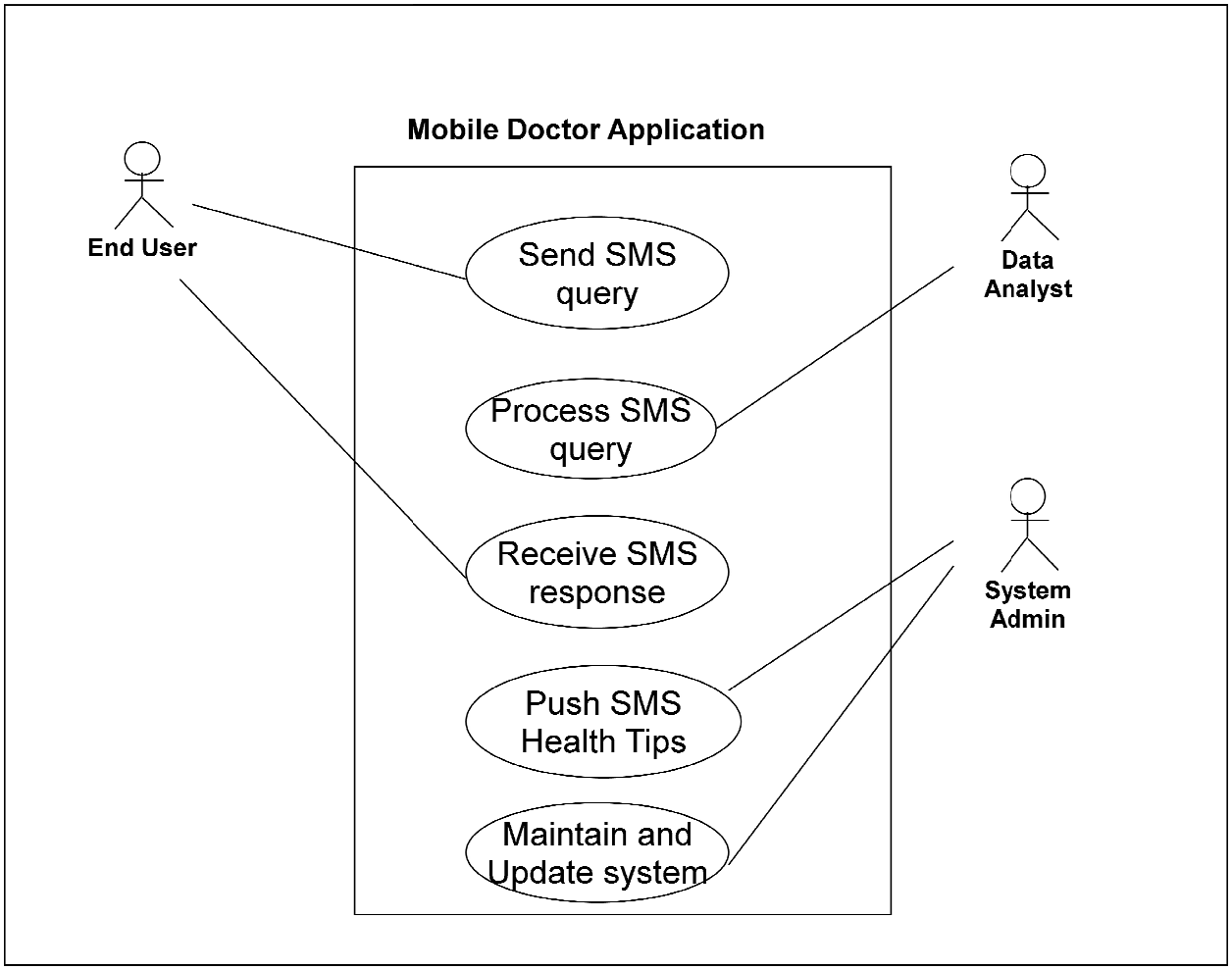

Figure 6: Use case diagram

b. Non Functional Requirements

The following non functional requirements are listed in order of their priority.

1. Availability / Reliability:

The nature of the Mobile Doctor Application is such that some of the clients that are using it could potentially be in great distress, i.e., Sick clients sending a text message to the Application. As such, there is a need for the system to be always available and hardly ever down. It should be available 24 hours a day, 7 days a week and should have redundancy set up in case of system crashes. A patient in pain needs to hear information that will help them as soon as possible. Therefore, our Mobile Doctor Application needs to be always available, i.e., responses are sent to a client whenever the client sends query. Also, routine maintenance and upgrades should be done as expeditiously as possible.

2. *Cost:*

Two types of costs should be taken into account, the cost to develop the application and the cost that the end-user spends using the application. While these two costs should both be low, it is even more important to ensure that the end-user cost be low. Two possible ways of implementing this cost are on a monthly basis, i.e., a flat monthly rate to use the service, or a charge per text to the Mobile Doctor number. If these costs are low, then more users will utilize the application.

3. Usability / User friendliness:

The Mobile Doctor application must be very easy for the client to use. A large number of users of this application will have little education and will be in the rural areas. As such, it must be as intuitive as possible. The Mobile Doctor App ensures this usability by allowing the client to simply type a text message with the illness or symptoms and receive information at their number.

4. *Performance:*

The Mobile doctor application should allow multiple users to use it at the same time. The system should be able to support several hundred users sending a text message to the servers at the same time. The servers should be able to handle these simultaneous requests. Also, the response time between sending a text message query and receiving a response should be less than 15 seconds assuming there are no network problems. This time period is not only dependent on the application, but on the mobile telephony network as well.

5. *Extensibility:*

While the basic logic of sending a query and receiving a response remains constant, other parts of the application could evolve and need new functionality added. For this to be done seamlessly, the software should be designed so that adding new functionality does not easily break existing functionality.

6. *Comprehensibility:*

In many cases, new developers must modify the application after a developer has left. The application should be designed in such a way that it is easy or intuitive for the new developer to understand what the previous developer was doing and where he left off. The code should also be well commented.

7. *Reusability:*

The system should be designed with reusable software components, i.e., classes and sub-systems can be "grouped" together such that a new application with similar functionality can reuse elements from this application.

8. *Maintainability*

The system should be designed using object-oriented principles that promote easier maintenance of the code base, e.g., reduced coupling and reduced complexity.

c. Data Requirements

We can group the data requirements into two categories: data input into the system and data output from the system:

Data Input:

- Human symptoms entered by user
- Animal symptoms entered by user
- Human disease names entered by user
- Animal disease names entered by user
- User input phrases describing condition
- Latest Medical Information
- Cost per SMS
- List of authorized database editors

Data Output:

- Remedy / First Aid information
- Prevention information
- “What not to do” information
- Daily Health Tips
- Legal disclaimers waiving liability
- SMS Invoice

- Database Entries

d. Timing Requirements

The major timing requirement in the Mobile application is the amount of time that elapses between a user sending an SMS message and receiving a response. We would like there to be a time lapse of less than 15 seconds between when the SMS is sent and a response received. This speed and time period is dependent on two things, the Mobile Doctor App and the Mobile Telephone Network. Obviously, we cannot control the network and its timing as this is handled by the telecom company and its infrastructure. In some instances, the network may be having glitches or communication issues that cause long delays in the time for a user to receive a response to their SMS query. However, we can control the speed of our Mobile Doctor Application by choosing efficient algorithms and data structures for parsing and searching. The speed of operation will be dependent on the speed of the search or parse algorithm.

e. Installation Requirements

The following are some of the key installation requirements of our system:

- Operating System – Linux Operating System – Open source and available.
- Database – Oracle RDMS
- JRE – Java Run Time environment
- Client device – Mobile phone
- Host application – Resides on Application server. An application server is a framework or set of servers that is dedicated for the efficient execution of programs, scripts or routines.

V. ARCHITECTURE

This section of the paper describes the proposed architecture of the system based on the aforementioned requirements. First, an overall description of the system is laid out, then different blueprints or architectures of the application are described. The first is the business blueprint which shows how the application functionality and data are grouped into different software components that make up the system. It is the architect's implementation-independent vision of how functionality is assigned to components. The Solution and Deployment Blueprint instantiate the Business Blueprint by describing implementation details and are presented together. They show how groups of functionality can be implemented by different technology solutions, computer platforms or even people.

DOMAIN DESCRIPTION – Overall Description of System

The Mobile Doctor Application is an SMS based application that enables end-users to receive remedy and first aid information. It relies on the end-user or customer sending a text message with their query to the Mobile Doctor telephone number. The query can be in form of a question, a symptom the patient is experiencing or a disease name. The architecture is a Pull SMS Architecture in which a client sends their query and receives a response from the Application Server and Database. When the message reaches the server side, it is parsed and redirected to the appropriate database from which it pulls the necessary information and sends it back to the customer. A charge is levied to the customer for this service either on a monthly basis or per SMS message. The application also has the ability to "push" daily health tips out to customers free of charge

a. AWAREness Business Blueprint

The AWAREness Business Blueprint [12] is an implementation-independent representation of how system functionality maps or is assigned to system components, i.e., how the different features that the system provides fit into components. Connectors between components depict the I/O flow between the components. They show how data from one component is used by a function in another component. The components in this case will be represented by classes.

Business Blueprint Derivation

Textual Representation of Business Blueprint Components:

COMPONENT	FUNCTION AND DATA
Disease	Function 1. Provide remedy pertaining to disease entered for humans 2. Provide remedy pertaining to disease entered for animals 3. Display prevention information as an option to the user. 4. Respond with "what not to do" pertaining to disease 5. Send and display the health tip of the day. 6. Ensure that response fits into one text
Symptom	Function 1. Provide remedy pertaining to symptoms entered for humans 2. Provide remedy pertaining to symptoms entered for animals 3. Respond with "what not to do" pertaining to symptom
Match	Function: 1. Map user input symptoms/disease to appropriate section in Database 2. Map symptoms entered to possible disease and vice versa
Parser	Function: 1. Parse verbose user input phrase for key words 2. Detect spelling errors entered by user 3. Detect input language and respond accordingly

Table 1: Textual Representation of Business Blueprint Components

Cost	Function: 1. Appropriately charge for remedy text message sent Data: 1. Charge rates per SMS
Legal	Function 1. Display legal disclaimer before every response can be read Data: 1. Legal Disclaimer

Table 1, cont: Textual Representation of Business Blueprint Components

Note: While these are the most of the functional features of the system, I have not included the few that are not customer oriented features but rather constraints or features pertaining only to the producers / production of the application, e.g., Data entries can be edited

I/O dependencies between components

The table below shows how input and output flow between components.

Components	*Functions and Data*
FROM: Symptom TO: Disease	• Ensure that response fits in one text **requires** Remedy information **from** Provide remedy pertaining to symptom for animals • Ensure that response fits in one text **requires** Remedy information **from** Provide remedy pertaining to symptom for humans • Ensure that response fits into one text **requires** "What not to do information" **from** Respond with "what not to do" for symptom

Table 2: I/O dependencies between components

UML Representation of Mobile Doctor Business Blueprint

Unified Modeling Language or UML is a standardized diagramming language in the area of Software Engineering. It is used to represent reusable software components, process flows, activities etc. In this case, UML is used to show the software components, their relevant functionality in the Mobile doctor application and how the components interact with each other as shown in Figure 7 below.

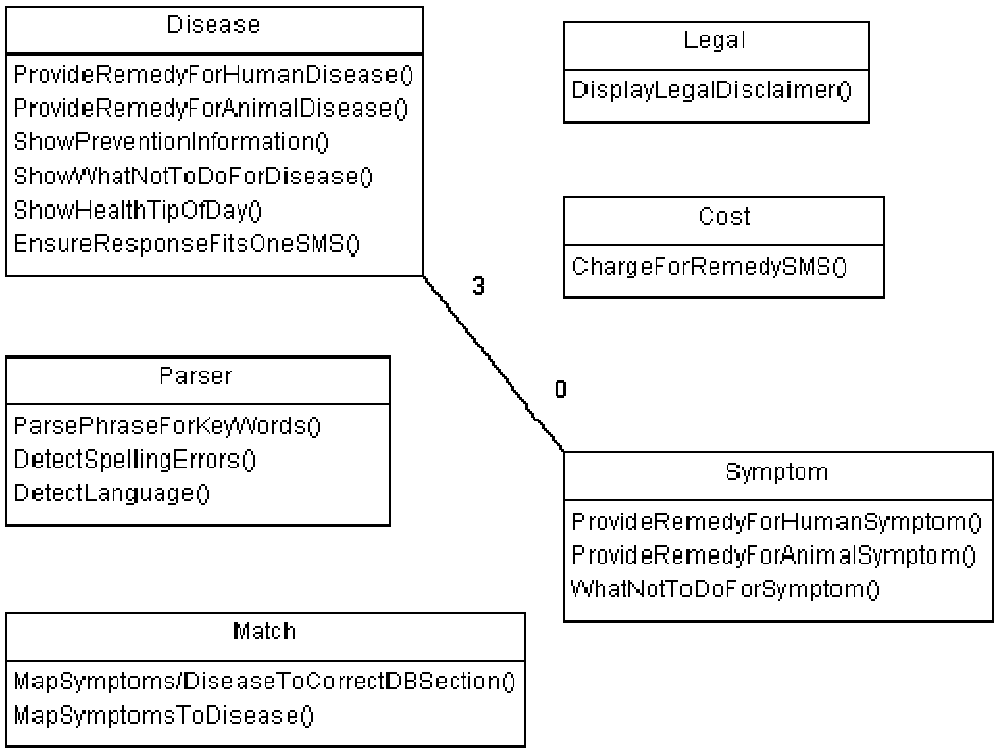

Figure 7: UML Representation of Mobile Doctor Business Blueprint

Derivation Plan and Rationale

When designing the system, the architect needs to come up with techniques or ways in which they can implement each of the non functional requirements. These techniques or rules are called heuristics in the AWAREness methodology [12] and layout specific strategies that can be implemented to meet a particular non-functional requirement. Table 3 below shows key non functional requirements, the heuristics that are applied to meet this constraint and the relevant reasoning behind the heuristics. Sometimes when trying to meet different priorities, tradeoffs need to be made, i.e., the heuristic applied to meet a priority of reusability such as having fatter components typically conflict heuristics for extensibility such as having thinner components. In such cases, heuristics associated with lower priority goals will typically be sacrificed for ones associated with higher priority goals. Potential conflicts and tradeoffs from Table 3 are discussed following the table.

1	Goal: Cost	
1.1	BB Heuristic: Group Based on implementation reality i.e. Cost component	Why: If we align functionality with components based on known COTS (Commercial Off The Shelf Solutions), we can save money by just purchasing that component as opposed to hiring developers to build that functionality. In our case, this is represented in the Cost component Priority Justification: Locals going to use our application typically are not wealthy people
2	Goal: Performance	
2.1	BB Heuristic: Reduce Data/Event Dependency i.e. Reduce component to component coupling from inputs/outputs (we have only 3 couples)	Why: Reduces communication bottlenecks. More inter-component communication channels will be slower than intra-component communication Priority Justification: Reduced component to component dependency means high performance means that the Mobile Doctor app can serve customers with response as within very short period of time.

Table 3: Derivation Plan and Rationale

3	Extensibility	
3.1	BB Heuristic: Isolate risk (isolate Cost component to facilitate enhancement)	Why: By isolating the functionality, this way, it can be updated or enhanced with minimal impact on other functions. Priority Justification: The way we charge customers may change from pay per SMS to a monthly fee or a completely different structure.
4	Comprehensibility	
4.1	BB Heuristic: Group based on Task Similarity i.e. Diseases component, Symptoms component	Why: By naming a component based on the functionality that it contains, it makes it interpretation of the code base a lot easier, e.g., Diseases component has functionality to provide remedy to diseases Priority Justification: One typically looks for a particular method in the same component that they have seen a similar method
5	Maintainability	
5.1	BB Heuristic: Reduced Coupling (Functions requiring Remedy Output are in components which have functions delivering that output)	Why: With less inter-component dependencies, there are less interfaces to manage and paths to follow. This makes it more maintainable. Priority Justification: Most logical way to reduce complexity and hence increase maintainability
5.2	BB Heuristic: Reduce Blueprint Complexity (number of components)	Why: Fewer components in a system are easier to maintain. Our architecture has only 5 components Priority Justification: Together with reducing coupling, reducing number of components clearly reduces complexity
5.3	BB: Heuristic: Reduce Class Complexity (reduce number of functions per component)	Why: Small components are easy to understand and modify Priority Justification: This heuristic will conflict with the heuristic to “Reduce Coupling” which advocates for more functions per component.
6	Reusability	
6.1	BB Heuristic: Reduce Data/Event Dependency (reduce component to component coupling)	Why: Less dependencies, more stand alone components that can be reused in other application Priority Justification: The need to expand on the application and use a number of the components again.

Table 3, cont: Derivation Plan and Rationale

Potential Conflict:

Heuristic 5.3 (Reduce class complexity) conflicts with the heuristic to reduce component-to-component coupling from inputs/outputs (2.1, 5.1, 6.1) because reducing class complexity seeks to reduce the number of services per BBC, reducing coupling will tend to increase the services per BBC.

Resolution:

One way to resolve this is to emphasize reducing coupling because it directly applies to the most important goal of the customers, "Performance" as well as maintainability and reusability goals

Coupling and Cohesion Metrics

Metrics showing the amount of coupling versus cohesion are needed to evaluate the satisfaction of stakeholder qualities. They describe the number of inputs/outputs between components, number of dependencies between components and the degree of cohesion. Metrics are shown in Table4 below

COMPONENT	INPUTS	OUTPUTS	Number of components that this component sends output to	Number of components that this component receives input from	Percentage of functions that send all input and receive all output from within component
Disease	5	4	0	1	16%
Symptom	3	2	1	0	0%
Match	2	1	0	0	0%
Parsing	3	1	0	0	0%
Cost	1	1	0	0	0%
Legal	0	1	0	0	0%

Table 4: Coupling and Cohesion Metrics

Size and Complexity Metrics

Size and Complexity Metrics help in the evaluation of stakeholder requirements by evaluating the size and complexity of components. They describe the number of functions per component, number of data elements per component, number of components per blueprint and the component complexity, which is given by "Number of data elements + number of functions + number of inputs and outputs from each function"

COMPONENT	Number of functions in component	Number of data elements in a component	Component Complexity
Disease	6	0	6+0+3 = 9
Symptom	3	0	3+0+0=3
Match	2	0	3+0+0=3
Parser	3	0	3+0+0=3
Cost	1	1	1+1+0=2
Legal	1	1	1+1+0=2

Table 5: Size and Complexity Metrics

Number of components: 6

Support for Applied Heuristics.

The component complexity in the size and complexity metrics display low numbers of complexity. This indicates a good level of comprehensibility for the new developers on the team. An architecture with fewer components or fewer data and functions in those components is clearly easier to understand.

On the Coupling and Cohesion Metrics, the fact that there are very low numbers for number of outputs sent to other components and inputs received from other components indicates low level of coupling. This low coupling benefits performance as well as the maintainability and reusability of the system. The lower percentages for "Percentage of functions that send all input and receive all output from within component" are more of an indication that this is not an I/O centric architecture.

b. AWAREness Solution and Deployment Blueprint

The AWAREness Solution Blueprint and Deployment Blueprint [12] instantiate the Business Blueprint by describing implementation details. Business Blueprint components are realized by people and technologies identified in the Solution and Deployment Blueprints. Two different deployment blueprint solutions are presented and the rationale behind each is discussed.

1. Deployment Blueprint #1 (DB1) – Manual Solution

In this case, we have a completely manual solution. People are playing the roles of Solution Components and also represent deployment components.

Satisfaction of domain functions by solutions:

SB SOLUTION COMPONENT	BB FUNCTION SATISFIED
System Admin	Function • Ensure response fits in one text • Charge per SMS for using service • Send and display the health tip of the day • Display legal disclaimer Data • Cost per SMS • Legal disclaimer

Table 6: Satisfaction of domain functions by solutions (DB1)

Database Admin	Functions • Provide remedy for Human disease name • Provide remedy for Animal disease name • Provide remedy for Human Symptoms • Provide remedy for Animal Symptoms • Display prevention information • Respond with “what not to do” pertaining to disease • Respond with “what not to do” pertaining to symptom
Data analysts	Functions • Parse User Input phrase for Key words • Detect spelling errors • Detect Language • Map symptoms to correct database section • Map symptoms to disease

Table 6, cont: Satisfaction of domain functions by solutions (DB1)

Allocation of solutions to deployment components

DB COMPONENT	SB COMPONENTS ALLOCATED TO DB COMPONENTS
System Admin	System Admin
Database Admin	Database Admin
Data Analyst	Data Analyst
Invoicing system	N/A
Legal advice / info	N/A
Medical Reference	N/A
Language guides	N/A
Thesaurus / Word references	N/A

Table 7: Allocation of solutions to deployment components

Graphical Depiction of Deployment Blueprint 1

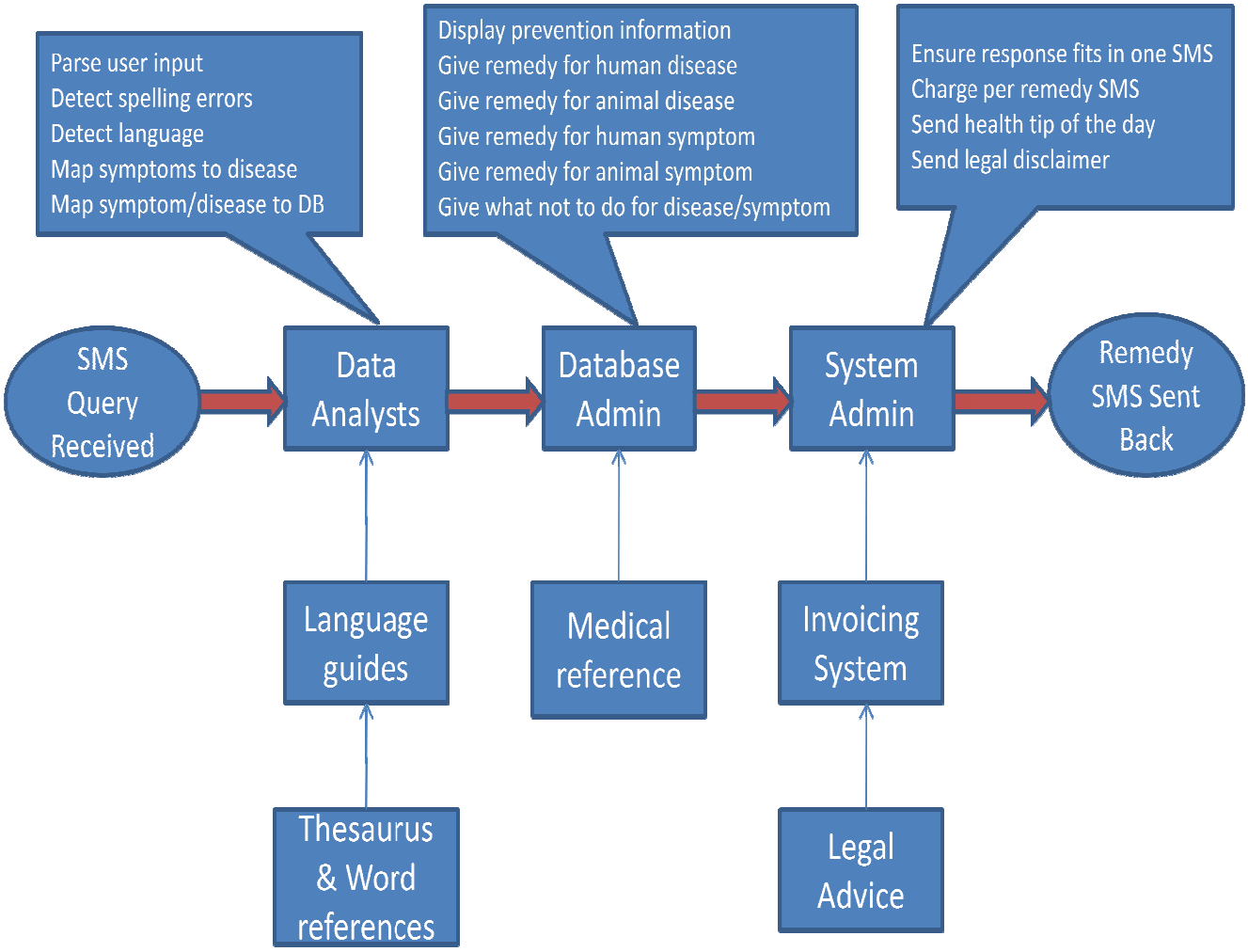

Figure 8: Graphical Depiction of Deployment Blueprint 1

Rationale:

Satisfaction of non-functional requirements / stakeholder qualities:

- Availability / Reliability – Because the system is based on people, we can have day and night shifts supporting the application. This means that at any particular time of the day or night, a customer query is responded to by the person on the other end.

- Cost – We are using a completely manual solution, with the players being the Data Analysts, the Database Administrator and the System Administrator
- Performance – The fact that we have 3 different skill sets of the Data Analysts, the Database Admin and the System Admin means that each can focus on a particular task that they are qualified to do well and thus complete it in a more efficient and quicker way than if it was one skill set trying to do all the tasks i.e. the data analyst does not have to worry about charging for the SMS or legal disclaimers but leaves that to one who will perform it faster.
- Extensibility – We can "swap out" Data Analysts who are familiar different languages whenever necessary
- Usability / User friendliness – The fact the system is completely manual opens up the possibility to live phone customer support. An end user with a problem can call in and talk to an actual person.
- Comprehensibility – Information concerning the inner workings of the system can easily be passed down by person to person teaching of new employees by current ones. This is possible because the system is made up of people. Contrast this with a new employee trying to read through volumes of manuals or deciphering a complex system on their own. That would be very time consuming and it would be difficult to dig out the most important information.

Design Inspired by:

- Domain Realities – Deployment Blueprint components closely mirror Business Blueprint components. The functionality in each Business Blueprint component has a mapping to a Deployment Blueprint component that encompasses most or all of the functionality in the Business Blueprint Component.

2. Deployment Blueprint # 2 (DB2)

Satisfaction of domain functions by solutions

SB Solution Component	BB Functions Satisfied
Oracle Database Software	Functions • Provide remedy pertaining to disease entered for humans • Provide remedy pertaining to disease entered for animals • Display prevention information as an option to the user. • Provide remedy pertaining to symptoms entered for humans • Provide remedy pertaining to symptoms entered for animals • Respond with "what not to do" pertaining to disease • Respond with "what not to do" pertaining to symptom
Abstract Billing Module	Function • Appropriately charge for remedy text message sent Data • Cost per SMS
Subscriber list-serve of numbers for daily tips	Functions • Send and display the health tip of the day.
Open source spell checker tools	Functions • Detect spelling errors entered by user
Mapping module	Functions • Map user input symptoms/disease to appropriate section in Database • Map symptoms entered to possible disease and vice versa
Google language translator	Functions • Detect input language and respond accordingly
Alert Boxes / Widgets	Function • Display Legal Disclaimer Data • Legal Disclaimer
Parsing algorithm module	Functions • Parse verbose user input phrase for key words • Ensure that response fits into one text

Table 8: Satisfaction of domain functions by solutions (DB2)

Allocation of solutions to deployment components

DB Component	SB Components Allocated to DB Component
Database Server	Oracle Database Software
Application Server	Abstract Billing Module
Application Server	Push service / Subscriber list-serve of numbers for daily tips
Application Server	Open source spell checker tools
Application Server	Mapping module
Application Server	Google language translator
Application Server	Alert Boxes / Widgets
Application Server	Server parsing algorithm module
Linux operating system	N/A
Redundant Server	N/A

Table 9: Allocation of solutions to deployment components

Graphical depiction of Deployment Blueprint #2

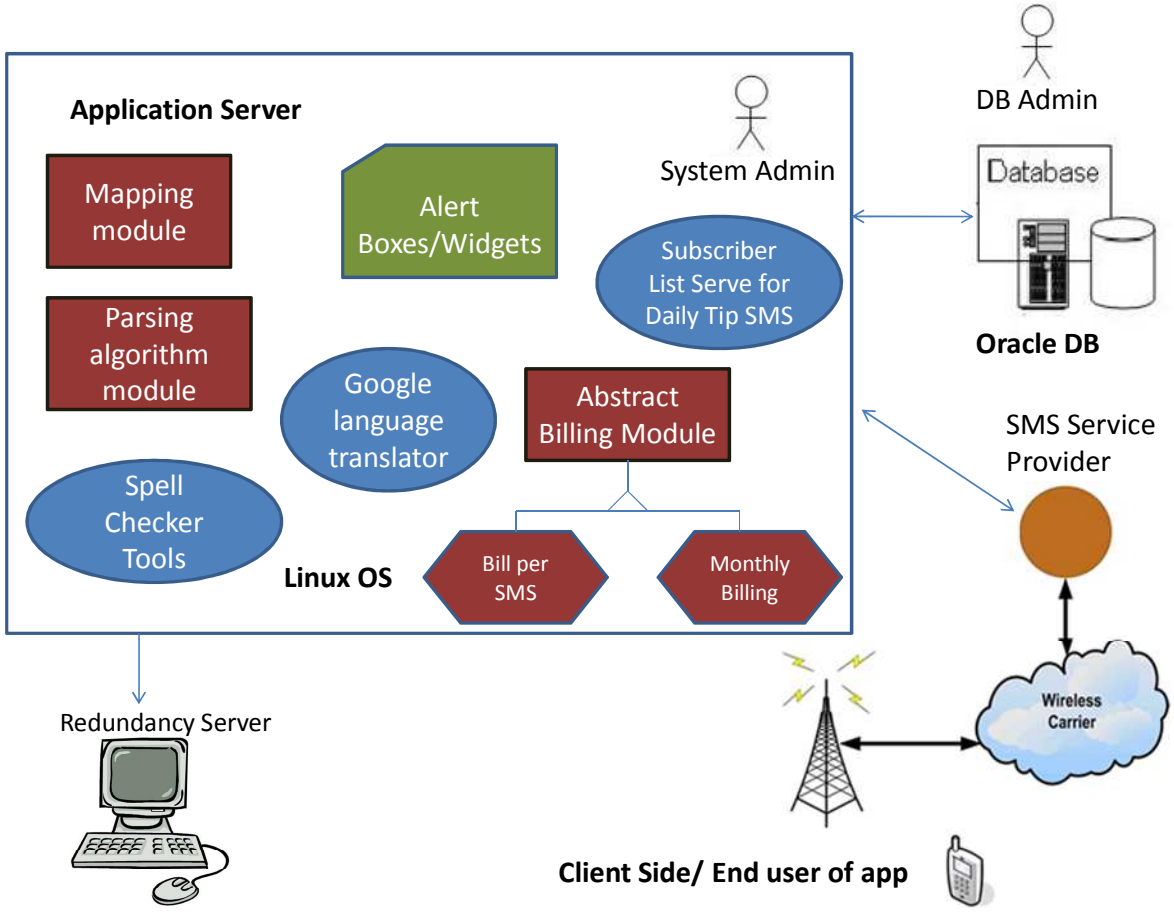

Figure 9: Graphical depiction of Deployment Blueprint #2

Rationale:

Satisfaction of non-functional requirements / stakeholder qualities:

- Cost – We are using a few open source software tools, e.g., Spell checker tools, translator software, Linux Operating System. These will be at no cost hence making the overall cost of designing the application cheaper and hence less cost is transferred over to the end-users.
- Reliability / Availability – Redundancy ensures that if the Application Server crashes, there is a back-up system available
- Performance – Use of efficient algorithms for parsing. This makes sorting and searching faster.
- Extensibility – An abstract billing module is used that has 2 different classes of billing that implement it. More implementations can be used and can be swapped out and the sub classes can be extended.
- Comprehensibility – The modules are named in an easy to understand manner and respective functionality is grouped under these modules and is hence easy to find.
- Reusability – Some modules can be used in other applications, e.g., the parsing algorithm module, billing module etc.
- Maintainability and Modularity – Keeping code in well defined, understandable modules, e.g., mapping module, keeps it modular and easily maintainable.

Architecture Inspirations:

- Client server architecture: This is the most intuitive way that this system should be designed, with the end users being the clients and the functionality / application modules on the server side to support the client. [9]
- Server redundancy: Redundancy to handle failures of the app server.

- Object oriented: Inheritance used for billing modules i.e. interfaces and classes that implement the interfaces
- Database Centric Architecture – Provides for storage of medical information.

VI. RESULTS AND ANALYSIS

a. Evaluation of Solution Blueprint Compliance

For each Solution blueprint, the AWAREness methodology describes an evaluation of how closely the technology used complies with the functions in the component. Thus, the AWAREness methodology offers a Component Function Coefficient to evaluate how closely the technology used complies to the functionality specified in the Business Blueprint for a respective component and a Component Data Coefficient, describing how well the technology in the Solution Blueprint complies with the data in a respective Business Blueprint component.

Deployment Blueprint 1:

CompFuncCoeff(c,t): Degree to which technology t complies to functions in component c

FuncCoeff(c,t) = Number of functions in c satisfied by t / Number of functions in c

FuncCoeff(Disease, Database Admin) = 4/6

FuncCoeff(Disease, System Admin) = 2/6

FuncCoeff(Symptom, Database Admin) = 3/3

FuncCoeff(Match, Data Analysts) = 2 /2

FuncCoeff(Parser, Data Analysts) = 3/3

FuncCoeff(Cost, System Admin) = 1/1

FuncCoeff(Legal, System Admin) = 1/1

CompDataCoeff(c,t): Degree to which technology t complies to data in component c

DataCoeff (c,t) = Set of Data in C registered to t /Set of Data defined in c

DataCoeff(Cost, System Admin) = 1/1

DataCoeff(Legal, System Admin) =1/1

Function Boundary Error

Technology	Max CompFuncCoeff, M	Sum of CompFuncCoeff, S	CompFuncBoundaryError, 1-M+S
System Admin	1/1	1/1 + 1/1 + 2/6 = 14/6	1 - 1/1 + 14/6 = 14/6
Database Admin	3/3	3/3 + 4/6 = 10/6	1 - 3/3 + 10/6= 10/6
Data Analysts	3/3	3/3 + 2/2 = 12/6	1 – 3/3 + 12/6= 12/6

Table 10: Function Boundary Error Table for DB1

Deployment Blueprint 2:

CompFuncCoeff(c,t): Degree to which technology t complies to functions in component c

FuncCoeff (c,t) = Number of functions in c satisfied by t / Number of functions in c

FuncCoeff (Disease, Oracle DB) = 4/6

FuncCoeff (Disease, List Serve) = 1/6

FuncCoeff (Disease, Parsing Module) = 1/6

FuncCoeff (Symptom, Oracle DB) =3 /3

FuncCoeff (Match, Mapping Module) =2/2

FuncCoeff (Parser, Parsing Module) = 1/3

FuncCoeff (Parser, Google Language Translator) = 1/3

FuncCoeff (Parser, Open source spell checker) =1/3

FuncCoeff (Cost, Abstract Billing Module) = 1/1

FuncCoeff (Legal, Alert Box/Widget) =1/1

CompDataCoeff(c,t): Degree to which technology t complies to data in component c

DataCoeff (c,t) = Set of Data in C registered to t /Set of Data defined in c

DataCoeff (Cost, Abstract Billing Module) = 1/1

DataCoeff (Legal, Alert Box/Widget)

Function Boundary Error

Technology	Max CompFuncCoeff, M	Sum of CompFuncCoeff, S	CompFuncBoundaryError, 1-M+S
Oracle DB	3/3	3/3 + 4/6 = 10/6	10/6
Abstract Billing Module	1/1	1/1	0
Subscriber List-Serve	1/6	1/6	6/6
Open source spell-checker	1/3	1/3	3/3
Mapping module	2/2	2/2	0
Google language translator	1/3	1/3	3/3
Alert Box / Widget	1/1	1/1	0
Parsing Algorithm Module	1/3	1/3 + 1/6 = 3/6	7/6

Table 11: Function Boundary Error Table for DB2

b. ATAM Quality Attribute Tree

Architecture Tradeoff Analysis Method (ATAM) is a process used early in the software development lifecycle for discovering tradeoffs and sensitivity points in the architecture. The purpose of the ATAM is to assess the consequences of architectural decisions in light of quality attribute requirements. [10] The ATAM tree for the Deployment Blueprints is presented following a brief discussion on the overall goodness of the system.

Definition and Calculation of Utility – Overall Goodness

The overall goodness of the system can measured by considering the priorities of the stakeholder goals and attaching a weighting factor to each of the goals based on priorities i.e. the highest priority goal has the highest weighting while the lowest priority goal has the lowest weighting. Next, multiply the weighting by a numerical value that indicates to what level the goal was satisfied. For example, my

performance goal which was priority four gets a weighting of 4 on a 1-8 scale since there were 8 priorities ranked in order of importance. Then the numerical value indicating satisfaction of performance might be high score of 100 if fewer than 2 couples exist in my overall system since reducing component to component coupling from inputs/outputs is aids the performance. Multiplying the weight (4) by the reusability score gives me my overall performance goodness. We do the same thing for the other goals and total all scores for all the goals each weighted differently. A higher overall score means more overall "goodness" of the system

ATAM Tree for Deployment Blueprint #1 – Manual Solution

- UTILITY
    - **Performance**
        - Increase the response time
            - *Objective*: User receives response in period of less than 60 minute
                - *Metric*: With a timer, count the number of seconds that elapse between user sending a message and receiving a response.
        - Maximize number of multiple users who can use the application simultaneously
            - *Objective*: Over 100 users can be querying the application at the same time.
                - *Metric*: Have both counter and timer functionality in the application. Observe and record statistics to see if the counter reaches 100 at an instant.
    - **Cost**
        - Minimize expenses on technology
            - *Objective*: Cut down on the number of expensive licenses for Software Products.
                - *Metric*: Have each employee report what licenses they own
            - *Objective*: Use manual solutions (people) wherever possible
                - *Metric*: Count number of man hours.
        - Maximize free resources
            - *Objective*: Use free dictionary, thesaurus, language guides when parsing user input information
                - *Metric*: Keep record of number of resources needed or used.

- **Reliability**

  - Ensure daily health tips sent out in timely fashion

    - *Objective*: Send daily tips out every 24 hours at a certain convenient time, e.g., 12pm
      - *Metric*: Keep record of time that health tip is sent out.

  - Ensure that someone is always there to support the system.
    - *Objective*: There should be night and day shift employees to manage the system at all times.
      - *Metric*: Count the number of hours the system has been manned by employees.
    - *Objective*: Have some experts available on call in case of major system crash
      - *Metric*: Count the number of emergency calls.

- **Extensibility**

  - Ensure replaceable Data Analyst role with different language speaking analyst.
    - *Objective*: Limit the job function of the analyst to translation tasks mainly
      - *Metric*: Count the number of other tasks the analyst is performing.

- **Comprehensibility**

  - Enable quick learning of system
    - Objective: One on one training of new employees since system is fully manual.
      - Metric: Weekly assessment of new employee's job performance
      - Metric: Test the new employees after training

- **Maintainability**

  - Ensure clearly defined job roles for Data Analyst, System Admin, Database Admin
    - Objective: Ensure that when a new task needs to be accomplished, there is no confusion over whose job it is to perform that task. That way all tasks get completed in timely fashion.
      - Metric: Using a matrix that indicates task assignment of task to person, observe if the tasks in the matrix have been owned by someone.

- **Usability**

  - Reduce end-user's interface complexity

    - *Objective*: Customer sends query simply by sending text message to Mobile Doctor phone number.

- *Metric*: Take customer survey asking how easy it is to use the application.

ATAM Tree for Deployment Blueprint #2

- UTILITY
  - **Performance**
    - Decrease time taken to decipher customer query
      - *Objective*: Use efficient sorting algorithms while searching for key words.
        - *Metric*: Measure the amount of time taken for mapping using Big O notation.
      - *Objective: Minimize data retrieval latency into Database.*
        - *Metric*: Measure the amount of time taken for pulling query out of Database.
  - **Cost**
    - Minimize Cost
      - Objective: Use open source software tools, e.g., translator, Linux OS
        - Metric: Keep track of percentage of open source software tools as compared to licensed tools.
      - *Objective*: Minimize cost of development
        - *Metric*: Determine initial hardware investment
        - *Metric:* Determine the initial investment in off-the-shelf software including widgets and database
        - *Metric:* Estimate the number of person-hours required to code-in house modules
      - *Objective:* Minimize cost of operation
        - Metric: Estimate the number of person-hours required to set the system up before use
        - Estimate the maintenance fees on off-the-shelf software.
  - **Reliability**
    - Minimize system blackouts
      - Objective: Install redundancy
        - Metric: Keep track of number of redundant servers
      - *Objective*: Use Distributed System.
        - *Metric*: Keep track of the number of servers that the application is running on.
  - **Extensibility**

- Enable pluggable billing service.
  - Objective: Maximize inheritance from abstract billing module
    - Metric: Number of properties inherited by Billing Module instantiation from ”Abstract Billing Module”
    - Metric: Number of methods inherited by billing module from “Abstract Billing Module”
  - Objective: Objective: Minimize coupling between “Billing” component and other BB Components
    - Metric: Measure coupling using “Number of dependencies between components”

- **Comprehensibility**

  - Enable easy understanding of system.
    - Objective: Use deployment solutions that match as much as possible BB components i.e. If BB component covers 3 functions, there should be a deployment solution that covers those functions and not many else.
      - Metric: Count the number of BB functions per component covered by DB
    - *Objective*: Reduce number of functions per component
      - *Metric*: Count the number of functions per component
    - *Objective*: Reduce total number of overall components to reduce complexity
      - *Metric*: Count the number of components used

- **Maintainability**

  - Code easily modifiable
    - *Objective*: Code should be written using object oriented based languages that allow functions to be logically grouped into easily manageable components
      - *Metric*: Count the number of components
      - *Metric*: Count the number of functions per component
  - Ensure code is version controlled
    - *Objective*: Successive revisions of code should be consistently versioned and stored in a software repository
      - *Metric*: Look through the different software version releases to ensure that they are versioned correctly.

- **Usability**

  - Ensure that the customer has easy-to-understand interaction with application
    - *Objective*: Customer sends query simply by sending text message to Mobile Doctor phone number.
      - *Metric*: Take customer survey asking how easy it is to use the application.

VII. CHALLENGES AND FUTURE WORK

One of the major challenges of the Mobile Doctor application is the legal aspect of it. The worst case scenario is that a patient follows the advice that he or she receives from the application and suffers severely or dies as a result. There needs to be enough legal consultation done to ensure that the producers of the application do not get sued by the patients or their families. Another challenge is to get enough credible doctors to agree to provide their expertise and health knowledge to build the database of the application. It could be risky for the doctor's career if there are a large number of reported negative results by users of the application. Another possible challenge could be that there would be some users who would just not want to consult their phone for medical information, but rather see a doctor face to face. This might be common in the rural areas where face-to-face interaction is how people relate. As a result, there might not be as many customers as hoped.

Future work on this application would involve extending the service to lend helpful information to crop farmers. Crop farmers are just as common as cattle farmers in the rural areas. They would benefit greatly too from SMS solutions concerning their crops that are not doing well. Another area that would be interesting to explore is the deployment of this application natively on the phone. This would require smarter phones to be used, e.g., iPhone, Android Phones. While Africa has not reached this stage yet, this could save the customers money in the long run, provide on-hand information and reduce network issues.

VIII. CONCLUSION

In this work, major challenges facing the African continent were presented and a mobile phone application technology to address those challenges was proposed. The creation of the Mobile Doctor Application, a text-message-based application that gives simple home remedy solutions for patients and cattle-keeping farmers, was discussed. The different types of requirements for the application were then represented, both textually and graphically. Based on the non-functional requirements, the features of the application were then grouped into components and presented in a business blueprint [12] depicted as a Unified Modeling Language diagram. From this, two possible deployment blueprints [12] were derived that would serve as deployment options, one completely manual and the other using technology. Evaluation of the architectures was then done using size, complexity, coupling, and cohesion metrics as well as an Architecture Tradeoff Analysis Method (ATAM) Tree.

BIBLIOGRAPHY

1. Laura Czerniewicz, “Mobile is My Soul: More about Cell Phones in the South of Africa”, e-Literate May 22 2010
2. Jenny C. Aker and Isaac M. Mbiti, Mobile Phones and Economic Development in Africa,, Center for Global Development, June 2010
3. Hash, 2007 African Mobile Phone Statistics, African Mobile Factobook, 2007
4. Massachusetts Institute of Technology, African Information Technology Intiative, AiTi, http://aiti.mit.edu/
5. John D. Sutter, “Mobile app developers tackle Africa’s biggest problems”, CNN Labs, April 2010, http://www.cnn.com/2010/TECH/04/12/africa.apps/index.html
6. Katherine Stapp, “Preventable Diseases Claim 11 Million Children Each Year”, Health, IPS News, July 4 2003
7. Jennifer Bryce, Cynthia Boschi-Pinto, Kenji Shibuya, Robert E Black and the WHO Child Health Epidemiology Reference Group, “WHO estimates the causes of death in children”, Lance, 2005
8. Paul E. Black, National Institute of Standards and Technology, big-O notation, November 20 2008
9. Len Bass, Paul Clements and Rick Kazman, Using Styles in System Design, Software Architecture in Practice, 1st Edition, Pgs 113-121
10. Rick Kazman, Mark Klein, Paul Clements, ATAM: Method for Architecture Evaluation, Software Engineering Institute, August 2000
11. Barber, K.S. and Graser, T.J., “AWAREness Requirements Engineering Methods,” course lecture notes from Requirements Engineering class , Fall Semester, 2009
12. Barber, K.S. and Graser, T.J., “AWAREness Architecture methods and artifacts,” course lecture notes from Software Architecture class, Spring Semester, 2009

VITA

Solomon Mugume Rugunda was born in Kampala, Uganda. After completing his high school at Kings College Budo, Kampala, Uganda, he attended college at the University of New Mexico from 2001 to 2005 and received the degree of Bachelor of Science in Electrical Engineering. During the following years, he was employed as a Software Engineer at National Oilwell Varco. In August 2008, he entered the Graduate School at The University of Texas at Austin

Permanent Address: 3101 Shoreline Dr, Apt 236

Austin, Texas 78728

This report was typed by the author.