10-25-2013

MobiMed: Framework for Rapid Application Development of Medical Mobile Apps

Frank Hernadez
hernandez.frank@gmail.com

DOI: 10.25148/etd.FI13112202

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

MOBIMED: FRAMEWORK FOR RAPID APPLICATION DEVELOPMENT OF MEDICAL MOBILE APPS

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Frank Hernandez

2013

To: Dean Amir Mirmiran
College of Engineering and Computing

This dissertation, written by Frank Hernandez, and entitled MobiMed: Framework for Rapid Application Development of Medical Mobile Apps, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Shu-Ching Chen

Jinpeng Wei

Wei Zeng

Armando Barreto

Sitharama S. Iyengar, Major Professor

Date of Defense: October 25, 2013

The dissertation of Frank Hernandez is approved.

Dean Amir Mirmiran
College of Engineering and Computing

Dean Lakshmi M. Reddi
University Graduate School

Florida International University, 2013

DEDICATION

To my amazing family.

ABSTRACT OF THE DISSERTATION

MOBIMED: FRAMEWORK FOR RAPID APPLICATION DEVELOPMENT OF MEDICAL MOBILE APPS

by

Frank Hernandez

Florida International University, 2013

Miami, Florida

Professor Sitharama S. Iyengar, Major Professor

In the medical field images obtained from high definition cameras and other medical imaging systems are an integral part of medical diagnosis. The analysis of these images are usually performed by the physicians who sometimes need to spend long hours reviewing the images before they are able to come up with a diagnosis and then decide on the course of action. In this dissertation we present a framework for a computer-aided analysis of medical imagery via the use of an expert system. While this problem has been discussed before, we will consider a system based on mobile devices.

Since the release of the iPhone on April 2003, the popularity of mobile devices has increased rapidly and our lives have become more reliant on them. This popularity and the ease of development of mobile applications has now made it possible to perform on these devices many of the image analyses that previously required a personal computer. All of this has opened the door to a whole new set of possibilities and freed the physicians from their reliance on their desktop machines.

The approach proposed in this dissertation aims to capitalize on these new found opportunities by providing a framework for analysis of medical images that physicians can utilize from their mobile devices thus remove their reliance on desktop computers. We also provide an expert system to aid in the analysis and advice on the selection of medical procedure. Finally, we also allow for other mobile applications to be

developed by providing a generic mobile application development framework that allows for access of other applications into the mobile domain.

In this dissertation we outline our work leading towards development of the proposed methodology and the remaining work needed to find a solution to the problem. In order to make this difficult problem tractable, we divide the problem into three parts: the development user interface modeling language and tooling, the creation of a game development modeling language and tooling, and the development of a generic mobile application framework. In order to make this problem more manageable, we will narrow down the initial scope to the hair transplant, and glaucoma domains.

TABLE OF CONTENTS

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURE PAGE

LIST OF ACRONYMS

CRF	Conditional Random Field
GA	Genetic Algorithm
GP	Genetic Programming
HE	Histogram Equalization
IDE	Integrated Development Environment
ICA	Independent Component Analysis
JDT	Java Development Tooling
LR	Logistic Regression
MRFs	Markov Random Fields
OLS	Ordinary Least Squares
OO	Object-Oriented
PCA	Principal Component Analysis
PRM	Patient Record Management Module
PVM	Patient Visit Management Module

CHAPTER 1

INTRODUCTION

In the medical field images obtained from high definition cameras and other medical imaging systems are an integral part of medical diagnosis. The analysis of these images are usually performed by the physicians who sometimes need to spend long hours reviewing the images before they are able to come up with a diagnosis and then decide on the course of action. In this dissertation we present a framework for a computer-aided analysis of medical imagery via the use of an expert system. While this problem has been discussed before, we will consider a system based on mobile devices.

Since the release of the iPhone on April 2003, the popularity of mobile devices has increased rapidly and our lives have become more reliant on them. This popularity and the ease of development of mobile applications has now made it possible to perform on these devices many of the image analyses that previously required a personal computer. All of this has opened the door to a whole new set of possibilities and freed the physicians from their reliance on their desktop machines.

The approach proposed in this dissertation aims to capitalize on these new found opportunities by providing a framework for analysis of medical images that physicians can utilize from their mobile devices thus remove their reliance on desktop computers. We also provide an expert system to aid in the analysis and advice on the selection of medical procedure. Finally, we also allow for other mobile applications to be

developed by providing a generic mobile application development framework that allows for access of other applications into the mobile domain.

1.1 Expert System

"An expert system is a computer program that embodies **expertise** about a particular domain, and can use **symbolic reasoning** techniques to solve problems in this domain; problems that would need the assistance of a human expert in the real world. An expert system should also be able to **explain** its conclusion" - Donald A. Waterman

As expressed by Waterman an Expert System (ES) must have a knowledge base (where the domain **expertise** is stored), an inference engine (so that it can perform **reasoning** to solve problems) and a user interface to allow for human intervention (for problems that it would require the assistance of a human expert).

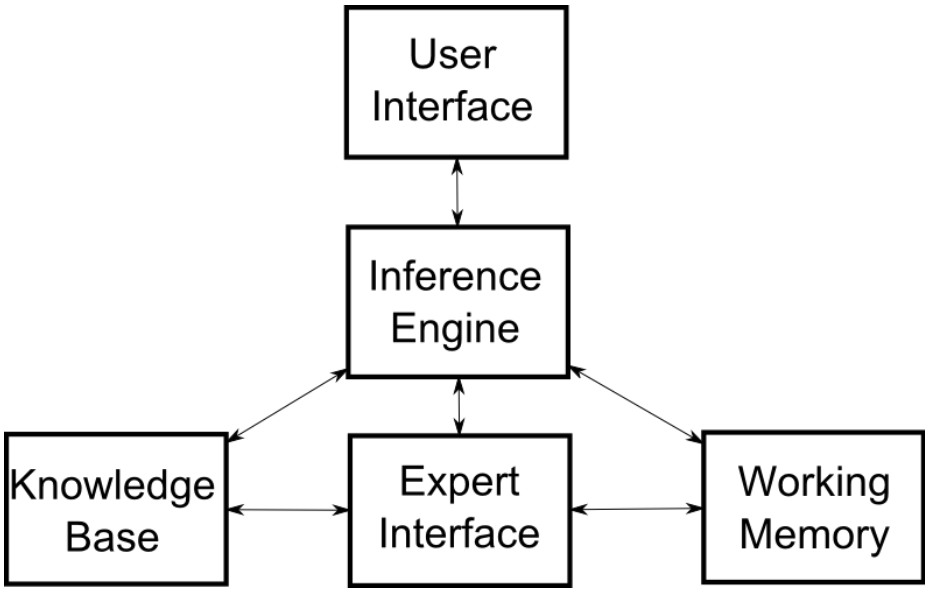


Figure 1.1: Components of a Basic Expert System

The **knowledge base** is where the knowledge of the expert system is stored. The knowledge contained in the knowledge base is usually gathered by the knowledge engineered and added for interpretation by the inference system. The most common method for representing the expert knowledge is in the form of rules. These rules are usually of the form:

$$IF\ condition(s)\ THEN\ action \tag{1.1}$$

The **knowledge base** may also contain facts about the domain. These facts can be descriptions of objects, as well as relationships between the objects. Using these facts and rules in the knowledge base the expert system can then proceed to determine whether assertions about objects are true or false as well as infer new properties of a given object.

The **inference engine** in an expert system uses the rules and facts stored in the knowledge base to validate assertions or provide suggestions to the user. The extent to which the inference engine can provide accurate assertions depends greatly on the type of expert system. Common types of expert systems include: fact-based expert systems, frame-based expert systems and semantic network expert systems. In the case of rule-based expert systems the success rate is directly proportional to the completeness of the knowledge base and can provide results that rival experts but cannot deal too well with uncertainty. On the other hand network and fuzzy-based expert systems tend to have a lower success rate but can deal especially well with uncertainty since the expert knowledge is expressed to allow more flexibility.

Both the knowledge base and the inference engine can be modified by the expert in order to improve the inference engine or the content of the knowledge base. In

order to allow for this, the expert system must provide an **expert interface**. This can usually range from shells, to graphical user interfaces, all the way to websites in the case of networked expert systems.

Similarly, the user interface allows the user to interact with the expert system. These interactions are usually in the form of queries through the inference engine. It is important to note that the user can either be a human or another computer system as it could be the case of a subsystem in a mobile application that requests suggestions to display to the user.

Finally, the working memory is used by the system to perform any active processing. It can also be used to hold temporary pertinent only to a scenario at hand which will be discarded once the query is completed.

1.2 Mobile Devices

Just last year in October (Figure 1.2) Flurry, the analytics for mobile devices, reported a total of 181 million active users using either Android or iOS mobile devices. With the population of the US at 313 million inhabitants this represents more than 57% of the population having either an Android or an iOS mobile device.

With so many owners of mobile devices and the large processing power that most of these have the number of opportunities to create new solutions like image analysis on the mobile devices have never been greater. The approach proposed in this dissertation aims to capitalize on these newly found opportunities by providing a framework for analysis of medical images that physicians can utilize from their mobile devices

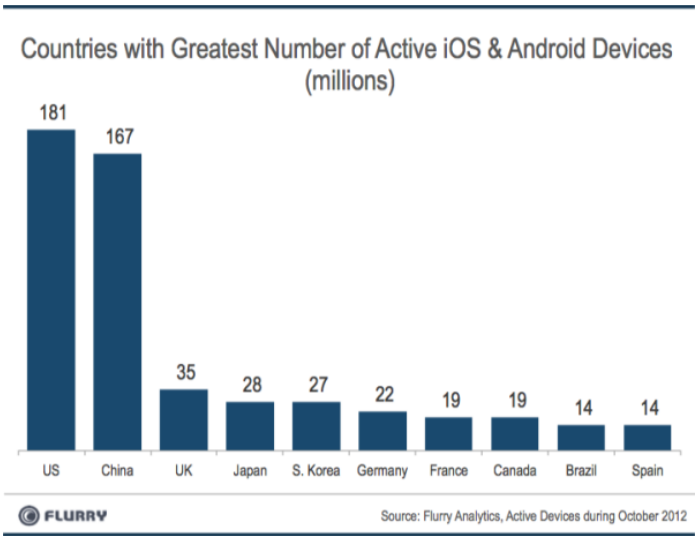


Figure 1.2: Flurry Active Users

thus remove their reliance on desktop computers. We also provide an expert system to aid in the analysis and advice on the selection of medical procedure. Finally, we also allow for other mobile applications to be developed by providing a generic mobile application development framework that allows for access of other applications into the mobile domain.

Figure 1.3: Mobile application for analysis of hair transplant images.

It is our hope to not only free the physicians from their dependency on desktop computers but also provide them with an improved way for diagnosis medical images. This will in turn reduce the time required of them to produce a diagnosis and improve their performance. However, a single mobile application is not going to make the impact in the problem that we would like. Hence, we are also developing a mobile framework for others to be able to easily develop medical mobile applications and increase the number of medical domains that can benefit from access to mobile devices.

CHAPTER 2

LITERATURE REVIEW

2.1 BACKGROUND

2.1.1 Feature Enhancement and Extraction

In most image processing tasks, feature enhancement is a key pre-processing step. Feature enhancement is the process of adjusting the images so that the results are more suitable for analysis. This is quite important for tasks such as image segmentation, image reconstruction, image registration, and feature tracking.

Once the feature enhancement pre-processing step is complete then we can proceed to perform the feature extraction step. Feature extraction is the step where a set of features or linear combination of them is chosen to represent each class. Once these classes are identified the feature step is usually linearly divided into a set of regions belonging to each distinct class. These are usually computed using a specialized version of the following generalized eigenvalue decomposition in Equation 2.1

$$M_W V = M_U V \Lambda, \tag{2.1}$$

where M_W is the metric matrix to be maximized and M_U is the metric matrix to be minimized [78, 59, 37].

One of the best known uses of 2.1 is linear discriminant analysis (LDA) [78, **?**, 82]. In the case of LDA, M_W is defined as the sample class scatter matrix, that is, the covariance of the class means,

$$M_W = S_B = \sum_{i=1}^{c} (\mu_i - \mu)(\mu i - \mu)^(T), \tag{2.2}$$

where c is the number of classes, μ_i the sample mean of class i and μ the global mean which includes the sample of all classes. According to [82] a first option for M_U is the within-class scatter matrix, that is, the sample mean covariance matrix of every class, and it is defined as

$$S_W = \frac{1}{n} \sum_{i=1}^{c} i, \tag{2.3}$$

in which, n is the number of samples, c is the number of classes and $\sum_{i=1}^{c} i$ is the covariance matrix of the sample class i.

Another linear feature extraction technique is principal component analysis (PCA) [120] and is specially useful for dimensionality reduction [63, 89, 93, 97, 119]. PCA transform is as presented in Equation 2.4

$$Y = U^T X, \tag{2.4}$$

where X is the centered original data matrix and each column contains the pixel value of one image vector x. U is the $m \times n$ orthogonal matrix, and its principal components

are found by recursively seeking out the directions of maximum data variance under the constant orthogonality.

Similar to PCA, independent component analysis (ICA) has also been extensively used as a common feature extraction technique. Although ICA was originally developed for separating mixed audio, it has seen a great success when applied to image analysis. ICA explains the data using statistically independent random vectors.

$$X \approx AS, \tag{2.5}$$

where X is the observed random data matrix, A is the mixing matrix and S is the matrix containing the statistically independent vectors.

Many empirical studies have shown that ICA outperforms PCA as a feature extraction method [7, 30, 61, 35, 83, 113, 121, 129]. Hence we have decided to consider the utilization of a modified independent component analysis algorithm as part of our proposed solution. Also, aside from a few exceptions [76, 75, 74], almost all PCA/ICA comparisons [6, 7, 30, 61, 35, 34, 83, 87, 107, 113, 121, 127, 129] have used FastICA or Infomax to perform ICA. Hence, our approach will begin with Infomax for feature extraction.

2.1.2 Image Enhancement

Image enhancement is the term used to refer to the sharpening or accentuation of image features such as edges, boundaries or contrast in order to make the image more useful for analysis. While a large set of surveys of the different methods for image

enhancements can be found in [2, 58, 31, 68, 94, 66, 4, 1, 3, 71]. This section covers some of the most common image enhancement methods.

One of the most important image enhancement methods is Histogram Equalization (HE). The basic idea behind HE lies on mapping the input image's intensity values so that resulting histogram will have a uniform distribution. The primary reason for HE success at image enhancements is because it expands the dynamic range of intensity values while flattening the histogram. This can sometimes introduce a significant change in brightness of an image and cause it to fail to properly enhance the image.

A typical histogram equalization for a given I with intensity levels in the range $[0, L-1]$, the probability density function $p(I_k)$ is defined as:

$$p(I_k) = \frac{n^k}{n}, \tag{2.6}$$

where n^k represents the number of times that the level I_k appears in the input image I for $k = 0, 1, ...L-1$ and n is the total number of samples in the input image. $p(I_k)$ is associated with the histogram of the input image which represents the number of pixels that have specific intensity I_k. We can define the cumulative density function based on 2.6 as

$$c(x) = \sum_{j=0}^{k} p(I_j), \tag{2.7}$$

where $I_k = x$ for $k = 0, 1, \ldots, L-1$. HE is a scheme that maps the input image int the entire dynamic range (I_0, I_{L-1}), by using the cumulative density function as a transform function.

Another set of methods which have seen special success when applied to medical images [114, 70] are the Laplacian pyramid and the wavelet-based methods. The essential idea behind wavelet is to iterate through the image while invertible smoothing at each iteration. During each iterations differencing operations are performed while maintaining the same number of pixels as the original image. Laine et al. [70] found improvement on the image enhancement of mammographies by applying wavelets with adaptive thresholding. This has seen a lot of success when applied to mammographies and digital radiography images[77, 114]. As presented by Burt et al. [15] the Laplacian pyramid is a sequence of error images $L_0, L_1, \ldots, L_N$ where each L_i is the difference between the two levels of the Gaussian pyramid. For $0 \leq 1 < N$ we define L_l as:

$$L_l = g_l - g_{l+1}, \tag{2.8}$$

where L is the prediction error, and g_l is the result of applying a given low-pass filter to the original image g_0. According to [15] since there is no image g_{N+1} to serve as a prediction image for g_N we say that $L_N = g_N$.

2.1.3 Edge Enhancement and Detection

In image processing, edge enhancement is a filter that enhances the edge contrast of an image in an attempt to improve its acuteness. The filter works by identifying sharp edge boundaries in the image and increasing the image contrast in the area immediately around the edge. This process serves to simplify the analysis of images by drastically reducing the amount of data to be processed while preserving useful structural information about the target boundaries. In the past 20 years much research

has been done in the area of edge detection [17, 8, 105, 56, 112, 117, 9, 5, 84, 11]. In this section we present the prominent algorithms grouped by edge detection problem categories.

The five groups of edge detectors presented in this section are: 1) edge strength, 2) model matching, 3) orientation analysis, 4) color edge, and 5) genetic programming edge detectors. The first group relies on the edge strength where this edge strength is computed locally by either a linear or non-linear operator. This group includes the Frei and Chen's edge consistency measure [36], Kisworo et al.'s instantaneous energy detector [64], Marr and Hildreth's Laplacian of Gaussian filter followed the zero-cross detection [80], Nivatia and Babu operators followed by thin thresholding [95], Ramponi's second-order Volterra filter [101], Robin's orientation mask [102], Rosen and Thurston 's method [104] for analysis of grayness and texture features, Shanmugan et al.'s optimun Fourier domain filter [109] and last but not least, Canny operators [18] which approach edge detection as a signal optimization problem. Canny operators maximize the signal-to-noise ration and minimize the localization error and the spurious responses. To this day Canny's algorithm still outperforms many of the newer edge detection algorithms [90].

Model matching edge detectors aim to discern edges based on filtering or matching of a given image of model. This group includes Brooks' method of fitting a plane or quadratic [12], Chen and Yang's method of fitting a B-spline with regularization [21], Ghosal and Mehrotra's method of fitting a generalized pulse edge or step edge using Zernike moments [43], Haralick's method which fits a cubic facet while detecting a zero-cross of the second directional derivative of the image [46], Nalwa and Binford's method of fitting a one-dimensional surface [91], and Tabatabai and Mitchell's method of fitting an ideal edge by preserving first, second, and third order moments [116].

Orientation analysis edge detection methods lay more emphasis in the orientation of the gradient than the emphasis intensity of the edge or shape in the image. This group include the works of Machuca and Gilbert's roof edge and step edge detectors [79], Martens method of using the Hermite transform [81], Gregson's operator which uses angular dispersion of gradients to detect edge ribbons [44], and Zuniga and Haralick's directional derivative operators [131].

For edge detection in color images, a number of approaches have been proposed from processing individual planes [19, 49] to true vector-based approaches [29, 48, 88, 108, 111, 118, 128, 27]. The main trade off between these two approaches lie in computational load vs algorithm performance. The computational load of computing edges on individual planes can be much smaller than that of computing edges on the color vector. However, vector-based approaches exploit the correlation between the color planes much more effectively than the computation on single planes. This is the reason why the large majority researchers have concentrated on the vector-based approaches. This group includes the works of Carron and Lamber [19] where they apply the Sobel operator to each component of the HSI space and combine the individual results a trade-off parameter between hue and intensity. Moghaddamzader et al. [88] describes a hybrid method which uses RGB and a hue-based derivative of the LAB space which the authors refer to as the HSI space in their paper. They also use a fuzzy membership function based on the intensity and saturation of the pixels to assign a relative value to the hue contrast. This value is then normalized and the Euclidian distance of RGB color vectors is averaged with the hue contrast measure to produce an edge detector based both on hue and intensity differences. Finally, Zenzo [27] proposed the calculation of the maximum Euclidean distance between the central pixel and all its neighbors, which is called the vector gradient.

The last group of edge detection approaches tackle the problem using Genetic Programming [67]. This group includes the works of Bolis et al. [10] who simulated an artificial ant to search for edges in an image and Ebner [32] who used for shift functions and other functions to approximate the Canny edge detector. Harris and Buxton [47] designed approximate response detectors in one-dimensional signals. Wang and Tan [122] used linear genetic programming to find binary image edges as terminals for binary images. Finally, Zhang and Rockett [130] evolved a 13 x 13 window filter for comparison with the Canny edge detector.

2.1.4 Expert System

To this day knowledge based systems or expert systems (ES) still remain an important tool to solving problems requiring expert knowledge [22, 33, 42, 55, 65, 92, 110]. Expert systems can be segregated into two main groups, those which rely on logical inference and those which rely on statistical inference.

Logical inference systems are usually referred to as rules based expert systems. In this kind of expert system the knowledge is represented as a set of rules (2.9) that are checked against a collection of knowledge about specific domain. These rules are then used to infer information about the situation. The two most common inference mechanisms used by these systems are forward chaining[**?**] and backward chaining[**?**].

$$IF\ condition(s)\ THEN\ action \tag{2.9}$$

In forward chaining the facts are first established and the new facts are deduced by working forward through the rules. On the other hand, in backward chaining, the

inference is performed backwards. In this mechanism, the inference begins from the THEN part of the rule and guesses at the conclusion. After a guess is found it then attempts to prove that the guess is correct by locating the rule for which the THEN condition matches the guess.

Statistical inference systems rely on information from a sample to learn characteristics about a population in order to provide a conclusion. One of the methods that is most frequently used in these systems is Bayes theorem 2.10.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.10}$$

This theorem states that for propositions A and B, $P(A)$ is the prior probability or initial degree of belief in A. $P(B|A)/P(B)$ represents the support B provides for A. Finally $P(A|B)$ is the degree of belief having accounted for B.

2.2 RELATED WORK

During the survey of the literature no expert system for the interpretation of medical images on mobile devices was found. However, over the past decade a large set in the medical domain have been developed. In this section we present a list of all the expert systems that were encountered to be relevant to the medical domain and that have been employed in the medical setting.

Expert System	Purpose
ABEL [99]	Provide extra consultation electrolyte and acid-base consultation
AI/GEN [86]	Provide diagnosis of deaf-blind syndromes
AI/RHEUM [62]	Provide non-rheumatologist physicians with diagnostic assistance in the area of rheumatology
ATTENDING [85]	Provide critique of a physician's preoperative plan for anesthetic management
CADUCEUS [100]	Provide medical diagnosis
CASNET [69]	Specialized medical diagnosis
CASNET/GLAUCOMA [69]	Specialized medical diagnosis in the area of glaucoma
EMYCIN [60]	Provide infectious disease consultation
GUIDON [23]	Provide medical knowledge and tutoring
HEME	Help physician in the analysis of patient information
INTELLIPATH	En expert system and image library for clinical pathologist
MYCIN [13]	Provide help identifying bacteria causing severe infections
XPLAIN [115]	Generates a consulting program by refinement from abstract goals

Table 2.1: Expert Systems Explored

2.2.1 Expert Systems

Out of the explored expert systems presented in Table 2.1 only CASNET[69] and INTELLIPATH had a minimal overlap with the proposed work. CASENET/GLAUCOM was design to diagnose the presence of glaucoma in patients. However, their proposed ES relies on textual information entered by the physician. Our proposed expert system on the other hand aims to infer this information from medial images of the eye and perform this diagnosis on a mobile device. On the other hand INTELLIPATH provided inference from medical images but it was focused only on cardiology.

2.2.2 Gamification Modeling Tools

As part of our proposed approach we present a generic framework for the generation of medical mobile applications. In order to fully support this goal this framework must support the creation of gamefied components for medical application. Since many of the generated mobile applications contain an aspect aimed at educating the patient on the procedure to be performed we foresee that these applications will have a need to have gamified or game-like functions. In order to accomplish this we proposed the development of a modeling language to model these game-like components. To accomplish this, our approach needs to abstract the complexity inherent to the game development process. Thus we propose Eberos GML2D as a modeling language to reduce this complexity by graphically abstracting some of these game concepts.

In this section, we discuss some of the existing works that share our goal: abstracting the complexity of game development through modeling. Prior work on reducing the complexity of the game development process includes tools like: **Unreal Kismet** [28, 16], **Game Maker** [45], and above all **SharpLudus** software factory [39, 41]. **SharpLudus**, the closest of these approaches to ours, is explored in detail at the end of this section.

Unreal Kismet

Unreal Kismet is a visual scripting system that can be used to create complex scripted sequences quickly and easily, with little programming knowledge [28, 16]. This provides a higher level of abstraction from the **Unreal Engine** [16], and provides

game designers with the ability to model their game's interactivity without much programming knowledge.

Unreal Kismet [28, 16] allows users to create objects and entities without making a distinction of graphical representation. Since **Unreal Kismet** is designed to model the interactivity of levels in the **Unreal Engine** [16], it has no means of modeling the graphical representations of game entities. Unlike **Unreal Kismet**, our approach **Eberos GML2D**, makes a clear distinction between sprites and animations, which are the representations of the graphical components. This allows the user of **Eberos GML2D** to model the appearance of the game as well as some of the behavior of the entities.

One aspect that **Eberos GML2D** can improve on is the concept of "combinations" used in **Unreal Kismet** [28, 16]. The approach referred as "combinations," includes gates and delays. The gates allow an impulse to either flow or stop the flow using "open", "close" and "toggle" modes. The delays are impulses, which are held for a set amount of time. One example of delays is for a shooting game, where, for every three seconds, a gun will fire toward the player. Our current version of **Eberos GML2D** provides little support for modeling the actions of the game entities, and thus can benefit from applying some of these concepts used in **Unreal Kismet**.

Game Maker

Game Maker [45] is a commercial visual game editor for game designers to develop games while not requiring programming knowledge. **Game Maker** is a very powerful tool that generates an interpreted language, called Game Maker Language

(**GML**) [45]. The list of functionalities includes multi-player network games, special sound effects, particle effects (e.g snow), advanced drawing function (e.g textured and colorized shapes), thus making it a full-blown game developer kit.

However, this power does not come without a price. Games generated using **Game Maker** are tightly coupled with the underlying implementation of **Game Maker** [45] and are produced as a single executable. Our approach, **Eberos GML2D**, models games regardless of underlying implementations; thus, the same game model can be used to generate a game for the computer as well as a game for a another platform without any changes to the model itself. With our current approach, we provide game developers with means of automating the generation of repetitive code, thus speeding up some of the game development process, something which cannot be accomplished by **Game Maker**.

The advanced functionalities of **Game Maker** could also be used as a guideline for future versions of **Eberos GML2D**. For example, they both share similar sprite functionality(e.g simple animation or composite animation) which can be used to improve our approach, that allows for automatic sprite partitioning and automatic collision detection. Their approach includes those functionalities as part of the sprite editor. In our case, the user must model this behavior.

SharpLudus

The closest work to **Eberos GML2D** is **SharpLudus**, a domain-specific language approach [39, 41]. It presents a domain-specific language called SharpLudus Game Modeling Language (**SLGML**), and is aimed at modeling video games in the 2D

adventure genre. **SharpLudus** can create a game with some restrictions without the need to code while generating .Net 2.0 C# code.

A restriction with **SLGML** is that it only models adventured games. At first glance, this decentralized approach of having multiple modeling languages: one for adventure games, one for racing games, one for shooting games, may seem the proper approach. However, the definition between adventure, racing or shooting games may be a blur, especially if one game designer may consider developing a hybrid design where different aspects from all these games are used to make a new game or even a new genre. In the previous example, using the decentralized approach mentioned, since the adventure DSL knows nothing about the racing DSL, such a game could not be developed. In addition, our belief is that the 2D gaming domain is specific enough to be expressed by a single language [39, 41].

At a superficial level, both languages share some common ideas. For example, they both have entities, **SLGML** being a lot more restricted. There can only exist one main character in **SLGML** restricting the use of simultaneous players. Another shared concept, is the support for sprites, which are common in game development. Sprite support in **SLGML** consists of a list of physical images, each of them having an independent delay and an option to loop the animation. In contrast, in **Eberos GML2D**, the physical link to the image file is created using **Sprite2D**, and animations are modeled with a logical representation using **SimpleAnimations** and **CompositeAnimations**. This means that unlike **SLGML**, where each frame must be loaded into memory when the animation is played [40]; in **Eberos GML2D**, the sprites are modeled with a single resource file, and animations are represented logically within this file.

SLGML is a very restricted domain-specific game modeling language. This restriction can be overcome by allowing the game designer to write additional code for customization. The constraints can explain why the author of **SLGML** needed to modify the game engine, allowing the game engine to adapt to the modeling language. Unlike **SLGML**, the code generated from **Eberos GML2D** models sits on top of current game engine or development libraries, requiring no modifications from these [38, 41].

2.2.3 RAD Frameworks (Mobile)

While there are a few frameworks for rapid application development such as PhoneGap, Appcelerator, appMobi, they don't offer any additional improvement to the development of medical mobile applications. These frameworks focus on reducing the amount of work required to develop mobile applications over multiple platforms. In order to accomplish this, developers program their application once and then automate the process of converting the existing codebase to platform-specific code. In this section we cove some of the more widely used RAD frameworks for mobile applications.

PhoneGap

PhoneGap is a mobile application development framework, based upon the open source Apache Cordova project. PhoneGap allows mobile application developers to write a mobile application once with HTML, CSS and JavaScript, and then deploy it to a wide range of mobile devices without losing the features of a native application.

Currently PhoneGap supports deployment to iOS, Android, Windows Phone, Blackberry, and webOS. This means that developers targeting each of those platform can effectively cut their development time by as much as 4/5 since they only develop the application once and PhoneGap handles the deployment to each individual platform.

PhoneGap achieves this reduction in effort by abstracting many of the features of mobile devices and creating JavaScript implementation which are exposed to the developers. PhoneGap currently supports the following features:

- Accelerometer
- Camera
- Contacts
- File
- Geolocation
- Media
- Network
- Network
- Notification (Alert)
- Notification (Sound)
- Notification (Vibration)
- Storage

Hence, PhoneGap reduces the time it would take to develop an application for each of the supported mobile platforms but does not have any noticeable reduction on the effort that it takes to developed a single application. On the other hand, our framework MobiMed aims at addressing that in the context of mobile medical applications by providing abstractions of features common to medical application which can be reused by developers and reduce the effort to develop a single application.

Appcelerator

The Appcelerator platform is yet another tool available for the rapid development of mobile applications. Just like PhoneGap it allows developers to program their mobile applications using web technolgies like HTLM 5, CSS, JavaScript which then is deployed natively to the individual mobile platforms. Currently Appcelerator supports deployment to Android, iOS, Blackberry and Tizen. Appcelerator also provides a set of features to developers such as analytics, performance management tools, and cloud services. Since these features do not have a significant impact in the development of mobile applications they will not be discussed.

appMobi

Much like the previous two frameworks appMobi targets the development of cross-platform applications by allowing developers to implements their mobile application using HTML5, JavaScript and CSS. Currently appMobi can deploy to IOS, Android,and Windows8 Phone and provides cross-platform APIs for:

- Push Notifications
- In-App Purchases
- Gamification
- Analytics and Updates
- Advertisement

Unlike the previous two frameworks the APIs provide resumable abstractions which can reduce the effort of developing a single mobile application as well as a cross-platform application.

CHAPTER 3

RESEARCH PROBLEM

In this chapter we motivate the problem to be investigated and present a detailed problem statement. Our research focus is in rapid application development for mobile devices. In the next section we provide the motivation for this research by emphasizing the need for mobile medical image analysis applications and highlighting the benefits to be gained from conducting the study. In Section 3.2 we concisely describe the problems to be investigated.

3.1 Motivation

The primary objective of the work presented here is to reduce the effort required to develop mobile medical application. Mobile application in general have proven to be more challenging and require more development effort than their desktop counter parts. Aside from the challenges inherent to mobile applications, developers of medical mobile applications face an additional set of challenges. In addition, medical mobile applications must also be FDA compliant as well HIPPA compliant. All this means additional work and time for developers.

It is our goal to develop a mobile framework for the rapid development of mobile applications. This framework will use a module-based approach and provide developers with ready-to-use components they can use during the development of their mobile medical applications. Through the use of modules we can not only provide

developers with error-free code but also with components which are FDA and HIPPA compliant. One of such components is a suggestion system which aims to increase the diagnosis efficiency of physicians while allowing them to enjoy the benefits of mobile technologies. In order to accomplish this our objective is to build a medical image interpretation powerful enough to satisfy all the diagnosis needs of the physician and light weight enough to run on a mobile device. In order to make this difficult problem tractable, we divide the problem into three parts: the development user interface modeling language and tooling, the creation a game development modeling language and tooling, and the development of a generic mobile application framework.

3.2 Problem Definition

The problem under investigation is the *formulation of a rapid development framework for the domain of medical mobile apps.* In this research proposal we outline our preliminary work leading towards development of the proposed methodology. The effort is divided into the following three sub-problems:

1. **User-Interface Graphical Modeling Language.** In order to reduce the effort required to develop mobile applications we aim at developing a graphical modeling language. This language will allow developers to create applications through a drag and drop interface and develop models for their interface. This models will then be translated into implementation artifacts such as code and XML files. Since this files are generated from the graphical models and the produces code has been thoroughly tested and is FDA compliant was aim to reduce some of the effort required to generate user interfaces for this mobile devices.

2. **Development of the suggestion system.** The objective of the medical image expert system is to correctly interpret the features of the medical imagery with minimal human interaction. As presented in Section 1.1 one of the key elements of an suggestion system is a knowledge base, where the knowledge is suitably presented so that it can be manipulated by the inference engine. With the help of our expert Dr. Nusbaum we have already begun narrowing down the common scenarios in the domain and have begun to identify the facts and rules for our expert system. We plan to build this knowledge base and populate it with the rules and facts needed to also provide recommendations as to what procedures must be performed for a given medical domain.

3. **Development of a graphical language for modeling serious games.** By using this language developers developers will be able to save time during the development of gamification components as well as serious games in their medical applications. Since the components are represented as models, they can be quickly validated and translated into development artifact such as source code and other initialization components. Since the code is generated from the models we can guarantee that the generated components will be error free and adhere to FDA standards.

4. **Development of the generic mobile application framework.** Due to the complexity of development frameworks available there is a lack of medical apps available in mobile markets. The proposed framework generalizes the components specific to the medical domain and makes it accessible to all developers through high levels of abstraction. The framework should provide means for interacting with the expert systems presented in this work to extract features from medical data. It should also provide for means to easily off load any operations too load intensive for a mobile device onto the cloud. Finally, it should

provide an extensible interface for easy addition of functionality not originally present in the framework.

CHAPTER 4

CONTRIBUTIONS

In this chapter we present the current state of our research. In the next section we identify the current state of the knowledge base for the expert system in Section 4.3. We present the current architecture and state of our proposed generic mobile application framework in Section 4.4.

4.1 UI Modeling Language

To further extend the benefits of our generic mobile application framework we have developed a graphical language to ease the creation of graphical user interface in mobile devices. This language abstracts the creation process by providing users with a graphical interface where the common concepts in mobile user interfaces has been abstracted into graphical components. These components themselves represent multiple classes or associations of operation at the code level. Our user interface modeling language thus reduces much of the work required to create graphical interfaces in mobile devices by allowing the user to model the user interface in a graphical environment. Our framework then takes this model as input and translates it into source code specific to the target mobile platform.

The language used by mobile platform varies greatly from platform to platform. For example, iPhone requires applications to be implemented in Objective C; in Android they must be implemented in Java while in BlackBerry they must be imple-

mented in C++. By using a graphical language for the user to model the user interface for his/her mobile application we not only reduce the effort to required to generate these user interface but also reduce the overall development time to generate mobile applications for created applications across multiple platform. We reduce this effort by using the graphical model as an intermediate representation which can then be translated into code for different platforms.

The translation of the models generated by our user interface modeling language into platform specific implementations (i.e. Objective C, Java, C++) is done through platform specific code generators. These generators take the model generated using our modeling language and translate each component into platform specific code. Since the code implementation that each of our graphical component is unique to each platform we separate the families of our modeling language by platform as: B-UIML, i-UIML, and A-UIML. Where B-UIML is the set of tools needed to translate models generated using our proposed modeling language into BlackBerry-specific code. i-UIML is the set of tools that translate the models to iPhone-specific code and A-UIML is the set for our Android devices. Figure 4.1 shows the look of our B-UIML interface with a sample of a BlackBerry interface. It was our goal to tackler some of the accidental difficulties associated with the development of mobile user interfaces. UIML attacks these accidental complexities with:

- **Representation Abstraction:** UIML abstracts away the components needed to develop UI application, and encapsulates all underlying repetitive code.
- **Graphical Representation:** UIML provides a graphical representation for developing UI for mobile apps and shows to the user only the necessary UI components needed for such apps.

- **Code Generation:** UIML generates verified code from user models. This aims at reducing bugs introduced by users during the implementation of repetitive code.

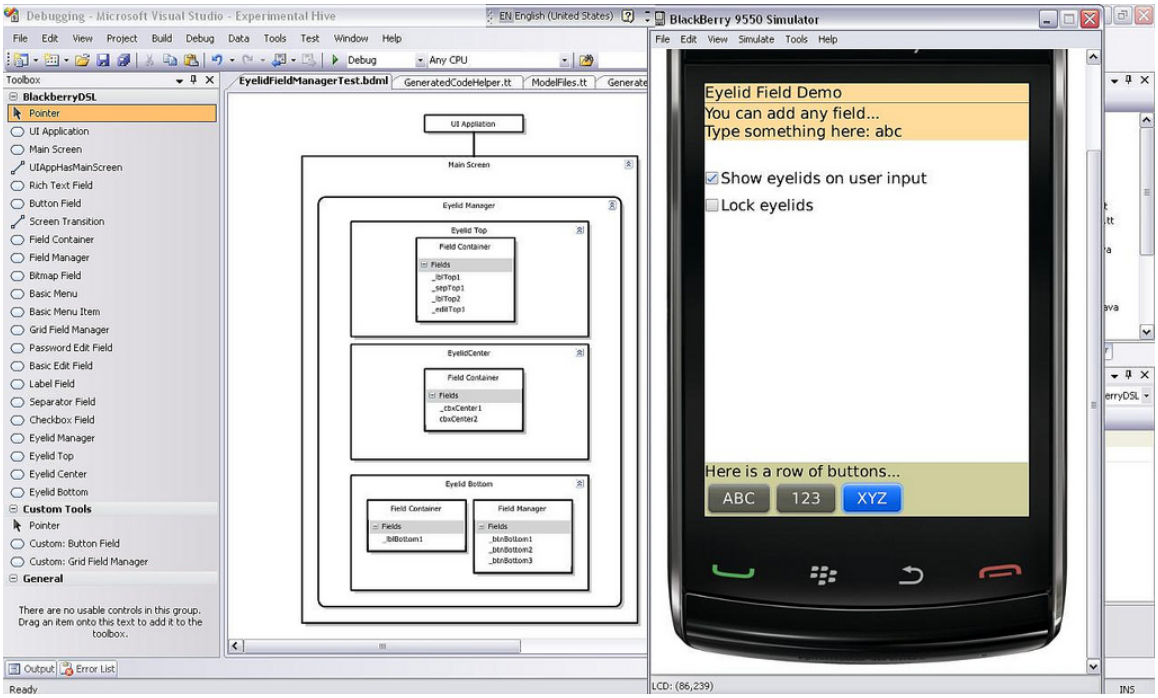


Figure 4.1: B-UIML - UI Modeling Tools

4.2 Eberos GML

In order to provide developers with tools that will allow them to easily develop all kinds of medical applications our framework must also facilitate the development of gamified components. Since we cannot predict all the needs an application may have we need to develop an approach that will allow future developers to easily implement these gamified mobile applications or even full blown medical mobile games. This is no easy challenge especially since over the past 30 years, and especially so the past 10 years, games have become so large and complex that they can no longer be developed by single-man teams [103, 20]. Even though this level of complexity is reduced by the

use of game engines, game development still remains a complex, time and resource-consuming task, taking teams from between two to three years to complete a single title [96]. We believe that the game industry could benefit from applying Model-Drive Development (**MDD**) approaches by raising the level of abstraction of the game-development process. We present a graphical Game Modeling Language (**Eberos GML2D**) that represents the game in an intuitive fashion. The abstraction provided by our graphical language also allows the models to be translated into game engine level code or code to be used by underlying libraries. This means that the same game can be produced for multiple platforms. Finally, we reduce the number of lines of code that must be written by game developers by automating the creation of repetitive code.

Game development is a multi-disciplinary process, bringing together: graphics, sounds, input, and networking. This means that developers must implement code to interact with the sound card, video card, input devices, and network devices as well as implement the code for the game itself. Most game engines already provide support for interacting with the platform's hardware, but few offer any kind of support for the development of the game code or logic.

Prior work on reducing the complexity of the game development process includes tools like: **Unreal Kismet** [28, 16], **Game Maker** [45], and **SharpLudus** software factory [39, 41]. **Unreal Kismet** is an editor that generates **Unreal Scripts** [28, 16]. It is designed for artists of the non-programing kind to develop their levels without the need to code. Unlike our proposed DSL, it limits the user to producing content only for the **Unreal Engine** [16]. **Game Maker** also provides users with a simpler method for developing games [45]; however, it does not provide any kind of reusable artifacts that could be deployed into other platforms or reused. **Eberos GML2D**

overcomes this by allowing for the transformation of the models into platform or game engine specific code. **SharpLudus** [39, 41], is the closer work of these three. While it provides the user with more control over the game modeling and target platforms like our **Eberos GML2D**, it is only limited to developing games in the domain of "Adventure Games," while our languages allows for the modeling of games from the entire 2D games domain.

In order to validate the expressiveness of our language, we decided to model two completely unrelated games. The first game, Pong, is the minimal game considered. Pong is composed of two paddles, one on each side of the screen, and a ball. The goal of the game is for each player to try and bounce the ball past the opposing player's paddle in order to score. The second game modeled is SpaceKatz, a title currently under development by the members of Florida International University's (**FIU**) SIG-Game, and it is a game of medium complexity consisting of menus, enemies and levels. The goal of the game is for the player to navigate his/her ship through the levels, avoiding obstacles and destroying the enemies along the way to beat a final boss. This game includes multiple menus, enemies, and levels, thus an obvious choice to demonstrate the current capabilities of our proposed language.

The remainder of this section is organized as follows: Section 4.2 describes the entire **Eberos GML2D**, its syntax, constructs, and meta-models. Section 5.2 presents the design of the experiments, our results and what findings we had. Finally we discuss these findings in Section 5.2.4.

Modeling a 2D Game

In this section we describe the requirements for a game modeling language, which was distilled after several discussions from domain experts and five years of game development experience of one of the authors. Then, we present the Eberos Game Modeling Language 2D (**Eberos GML2D**), which is our proposed language to meet our stated requirements. In particular, we have identified the following requirements for the game modeling language:

Simplicity: It must be simple and intuitive.

Platform-independent: It must be independent of any underlying game platforms.

Library/Game Engine-independent: It must be independent of game engines and/or development libraries.

Expressiveness: It must be able to model a large majority of game development scenarios.

Sprite and Animations

A sprite in the 2D game development domain refers to a two-dimensional image that contains the graphic representation of a single item in the game. Sprite sheets contain multiple consecutive smaller sprites that can be used by the game developer to create the illusion like movement or explosions, among other illusions. A 2D game without any images or movement can be very dull at best, so any game modeling language must provide support for modeling graphical behavior.

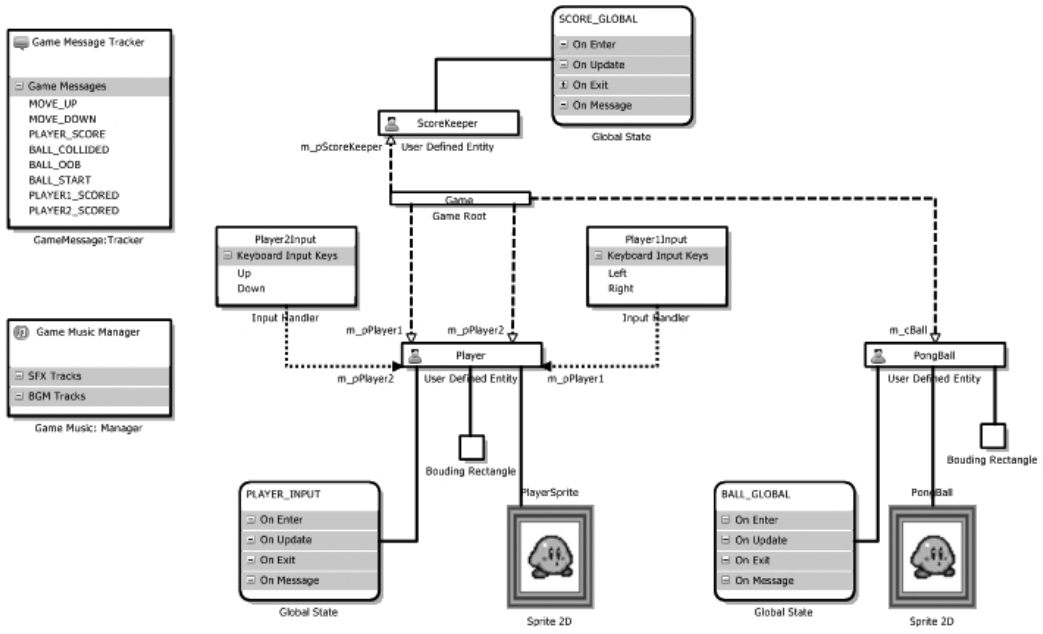
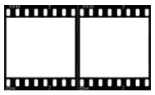


Figure 4.2: Eberos GML2D Model for the Game of Pong.

In our language, a sprite is represented by the **Sprite2D** construct. This contains information about the initial location in the file system of the image or resource file. A **Sprite2D** also contains information about the height, width, and X and Y positions on the screen of each individual sprite. The size and position information are used when a sprite has no specified animations. If the sprite has an animation, then the size and position information of the current animation replaces the information specified in the Sprite2D.

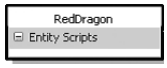
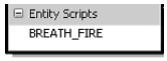
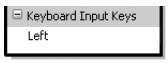
Animations are used to graphically represent the actions that are performed by game entities on the screen, such as swimming. In **Eberos GML2D**, each **Sprite2D** can have zero or more animations which can be of one of two kinds: **SimpleAnimations** or **CompositeAnimations**. In **Eberos GML2D**, **SimpleAnimations** are composed of uniform frames, ordered from left to right, all together representing a graphical action taken by a game entity. A **SimpleAnimation** contains information

Table 4.1: Sprite and Animation Terminals

SimpleAnimation	
CompositeAnimation	
Frame2D	
Sprite2D	

about the location on the resource image of the first frame of the animation, the delay of each frame and the total number of frames in the animation. **CompositeAnimations** give more freedom to the modeler when it comes to modeling a sprite's animations by allowing the specification of the animations on a frame-by-frame basis. In our language, animation frames are modeled using the **Frame2D** construct and only **CompositeAnimations** are composed of one or more **Frame2Ds**. **Frame2Ds** contain information about the position on the resource file and the size of a single frame of the animation. **CompositeAnimations** differ from **SimpleAnimations** in that they can be composed of frames of different sizes, represented by **Frame2Ds**.

Table 4.2: Entity Terminals

UserDefinedEntity	
ActionScript:Entity	
EntityPool	
InputHandler	
Keyboard:Key	

Entities

Every game is composed of entities of one kind or another. While sprites are graphical representations of game components, entities are the program representation of these components. These entities can represent players, items, enemies, and menus, among other components. The same way that games revolve around the concept of entities so does our language.

Currently, **Eberos GML2D** supports the modeling of **UserDefinedEntities** and **EntityPools**. **UserDefinedEntities** allow the modeler to model custom game entities, and are represented using the **UserDefineEntity** construct. In our lan-

guage, entities can contain zero or one sprite, thus allowing the modeler to represent both graphical and non-graphical entities. While graphical entities may represent players, and enemies, non-graphical entities can be used to model those entities that cannot be seen by the game player, such as triggers or area boundaries. In our language, **UserDefinedEntities** can be composed of zero or more entity references; this allows for the definition of atomic as well as composite entities while modeling the games.

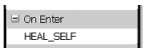
Entities in a game can either be controlled by the player **(PC)** or by the program, as in the case of non-playable characters **(NPC)**. In **Eberos GML2D**, **InputHandlers** are used to model the user's input handling logic. Each entity reference that can be controlled by the player's input will have an **InputHandler** attached to it in the model. The current version of the language supports only the modeling of keyboard input, but the ability to model mouse input and console controller input are currently being explored. Keyboard input is modeled by providing the **InputHandler** with a **Keyboard:Key** for each key to handle. Each **Keyboard:Key** can only map to a single key and is selected from a keyboard key enumeration containing every key in the keyboard. These allow the user to specify which key the **InputHandler** will handle, as well as what **GameMessage** the input handler will send the controlled entity when that key is affected (**GameMessages** will be covered later in this section). **Keyboard:Keys** can be modeled to detect one of four possible key states; **DOWN**, **RELEASED**, **WAS_PRESSED**, and **IS_HOLDING**. This allows the user to model a specific behavior, depending on the state of the key. One might want the player character to move left for as long as the "Left" key is pressed (**DOWN**) but might only want the player's character to attack every time the "Space Bar" is pressed (**WAS_PRESSED**). The means for modeling the behavior of program-controlled entities are described in the following section.

Logic

Having the ability to model the game logic for game entities was one of the main motivations for developing **Eberos GML2D**. While working on a previous project, we realized that a lot of the code used for modeling our game entities' logic was being constantly reused, and only a small section was modified slightly for each specific entity. This led us to develop a language for modeling this logic, which in turn led us to explore further and see how much of the game could be modeled, thus giving birth to **Eberos GML2D**. The current version of **Eberos GML2D** supports modeling of Moor finite state machines with one concurrent level of execution. This will not remain for long, as we continue to expand our language to allow for modeling of goal-based agents [106], as well as other kinds of artificial intelligence (**AI**) agents.

In our current version of the language, game entities may possess at most one **GlobalState**, and at most one **State** [14]; these represent the initial states of the FSMs for this entity type. In **Eberos GML2D**, finite state machines are modeled by connecting states together via the use of a **TransitionMessage**. A **TransitionMessage** represents the message that triggers a transition from one state to another. Each entity can have at most one global state machine, and at most one current state machine. In our language, transitions can only occur between states of the same type. **States** can only transition to **States**, and **GlobalStates** can only transition to **GlobalStates**. In **Eberos GML2D**, **States** are used to model the logic of a game entity, and can also be used to model the AI or behavior of computer-controlled game entities. **GlobalStates** are similar in behavior to **States**, but instead they model behavior that may occur during the execution of any state of the entity.

Table 4.3: Logic Terminals

State / GlobalState	
Action Script: On Enter	
Action Script: On Update	
Action Script: On Exit	
Action Script: Message	

Our language also allows the user to insert previously-generated scripts to further increase the state's logic; these are provided in the form of **ActionScripts**. **ActionScripts** come in four varieties: **ActionScript: OnEnter**, **ActionScript: OnUpdate**, **ActionScript: OnExit**, and **ActionScript: OnMessage**. Each **ActionScript** contains information about the script's ID, the script's import order and the script's location in the file system. The import order of a script represents the location where the script should be inserted in the final code in relation to other inserted scripts. Each of these scripts adds code to the state's respective section

Table 4.4: Collision Terminals

BoundingRectangle	
BoundingCircle	

OnEnter, **OnUpdate**, **OnExit**, and **OnMessage**, and is assumed to have been validated by an external tool. From all of the scripts, **ActionScript: OnMessage** is the only one that is handled in a different fashion; this represent snippets of code that will only be executed when the specified **GameMessage** is received.

ActionScripts allow the modeler to add behavior to the entity logic that can not be modeled with the current version of **Eberos GML2D**. **ActionScripts** are expected to be valid code for the underlaying platform on which the final generated code will be running. The current nature of these scripts both add expressiveness and restricts our language. Since the **ActionScript** are directly copied over during the translation of the model, this means that the game model can only be translated to a specific platform, thus reducing the number of platforms to which the model can be translated. At the same time, this script allows for support of behavior that could not otherwise be expressed with the current version of our language.

Collision Detection

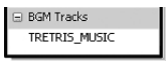
Collision detection refers to the act of detecting whether or not two or more game entities have come into contact (collide) with one another. There are a few methods for handling collision detection in 2D games: bounding boxes, bounding circles, and pixel-level collision, just to mention a few of the most common ones. **Eberos GML2D** supports modeling of two of these, bounding circles and bounding rectangles (bounding boxes). In our language, entities may contain zero or one bounding object which can either be a **BoundingRectangle** or a **BoundingCircle**.

BoundingRectangles are used to model collision detection between entity boxes. A **BoundingRectangle** represents a bounding box for a game entity, which is used in the game to detect when a collision has occurred. Currently, the user can specify both the width and the height of each entity's bounding rectangle. Similarly, **BoundingCircles** are used to model collision detection between circles. The user specifies the radius of the circle for the entity's collision detection.

Game Controllers

Game controllers allow the user to model game managers and similar control units, such as music managers and message managers. The **GameRoot** construct represents the entry point into the game. All games must originate at the game root; this represents the Main in many programs. The **GameMessage:Tracker** contains all messages available in the game. Any message handled or triggered by game entities must be registered with the **GameMessage:Tracker**; this ensures that all messages exist in the final generated game code. **GameMessages** represent a single type of

Table 4.5: Game Controller Terminals

GameRoot	
GameMessage:Tracker	
GameMessage	
GameMusic:Manager	
SFXTrack	
BGMTrack	

message in the game, usually an enum or value. Games can be composed of multiple messages, each representing a kind of information to be sent to or handled by game entities. Last but not least, we have the **GameMusic:Manager**; it allows the **Eberos GML2D** user to model all the music and sound effects in his/her game. This allows the user to specify the sound resource that is needed by allowing him/her to specify the location in the operating system. Upon translation, all the resources are placed in a local directory, and their correct path is written along with the code generation. This automation has been shown to reduce the number of code errors

occurring from incorrect resource paths. The user can model sound effect tracks using the **SFXTrack** construct, which allows him/her to set the in-game ID of the track and graphically specify the resource file. Sound effect tracks are used for short sounds used inside the game, a gunshot or an explosion. Similarly, longer music tracks can be modeled with the **BGMTrack** construct. A **BGMTrack** represents a single background music track. These tracks are used for playing songs or similar scenic music. **BGMTracks** are much longer in duration than **SFXTracks**.

Experimental Evaluation

In this section, we describe the experiments we performed to evaluate our language **Eberos GML2D**. We first present the design of our two experiments, then the results, followed by a discussion of our findings. Since our intent is to reduce the amount of both time and work required to develop 2D games, we use two metrics in our experiments: time and lines of code (**LOC**) required to develop each game with and without **Eberos GML2D**. We used Microsoft's XNA game engine for developing the games from scratch. The generated models were also translated into XNA Code as well.

Procedure and Scenarios

In order to evaluate **Eberos GML2D**, we developed each of the following game specifications first without using our approach, and then a second time using our DSL. We then timed both entire processes from start to completion. The second metric we used was the lines of code (**LOC**) of each version of the game, DSL vs.

Non-DSL. Since SpaceKatz (Game Specification 2) is an attempt to model a game being developed by the students of SIG-Games at Florida International University (**FIU**), we decided to use their existing project as a comparison instead of writing this game from scratch ourselves. We believe this gives us a more accurate set of data by comparing our approach with a real ongoing game-development process.

Game Specification 1 - Pong: Pong is as simple a 2D game as they come; it consists of two paddles (one for each player located at each end of the screen) and a ball. The ball bounces off the walls and the player's paddle. The goal of the game is to get the ball past the opponent's paddle in order to score.

Game Specification 2 - SpaceKatz: SpaceKatz is a 'Shoot 'em Up' style of game, where the player must pilot through asteroid fields while destroying enemies in the process. This game consists of one main screen with the game logo. Following the logo screen, the player is presented with a menu where he/she can select what options to set or directly start the game. After selecting to start the game, the player is presented with a menu to choose the level or area to play. Once an area is selected the game begins, and the player is presented with the spaceship to pilot. A player can move in all four directions by pressing the keyboard keys (Left, Right, Up, Down); he/she can also shoot missiles by pressing the "Space Bar" key on the keyboard. The game ends when the player loses all of his/her lives.

Experiment Setup

In this section we present the set up for both or our experiments. The results in lines of code represents the number of lines of code in XNA/C# required to complete both the game by hand and the game using **Eberos GML2D**.

Set Up Experiment 1 - Pong: First the game of Pong was implemented by hand, then it was modeled and implemented, both by the main author of this paper.

Set Up Experiment 2 - SpaceKatz: For the experiment of SpaceKatz, we used an existing project from SIG-Games. We gathered the information of lines of code and time taken from this existing project then we preceded to model the current state of this game and program it using our language. In the case of SpaceKatz the game without our proposed language was developed by the members of SIG-Games while the game implemented using **Eberos GML2D** was written by the main author of this paper.

Note: We feel it is important to note that SIG-Games is a group dedicated to training students to become game developers, and that past members have worked on AAA titles such as **Bioshock 2**.

Results

In order to perform the experiment, we developed a graphical editor for **Eberos GML2D** using the Microsoft DSL Tools [26]. In order to show the efforts saved using our approach compared to implementing directly, we have summarized our results in

Table 5.3 and Table 5.4. We also implemented a code generator to transform **Eberos GML2D** models into Microsoft XNA code. The numbers given represent the two metrics we used when evaluating our language: time and lines of code (**LOC**).

Table 4.6: Effort for creation of Pong using Eberos GML2D vs. manually developed

Game: Pong	**Eberos GML2D**	**Manually Developed**
Dev. Time (Hours)	0.13 Modeling + 0.38 Implementing = 0.51 total	0.56 Implementing
Effort (LOC)	92 User Gen. + 74 XNA Gen. + 1870 Auto Gen. = 2036 total	131 User Generated + 74 XNA Generated = 205 total

For each game explored, we show the actual effort of directly implementing the game, and the actual effort of implementing the game using our language. In Pong, we could reduce 39 (131 - 92) lines of code (**LOC**) written by the user and, as a result, 29% (39/131) of the amount of work required by the user was reduced. Table 5.3 also shows a savings of 3 (34 - 31) minutes, a 8.82% (3/34) savings on the time required to develop the game by using **Eberos GML2D** as opposed to implementing it without it. We also present the results of applying our approach to SpaceKatz, a more complex game than Pong. Using our approach as seen in Table 5.4, we were able to reduce 3303 (3822 - 519) lines of code written by the user, and save 86.4% (3303/3822) of the amount lines of code as opposed to implementing the game directly. There was also a significant reduction of 24.67 (30 - 5.33) hours required, a savings of 82.3% (24.67/30) of the time spent creating the game by using our approach as opposed to implementing the game without it.

Table 4.7: Effort for creation of SpaceKatz using Eberos GML2D vs. manually developed

Game: SpaceKatz	**Eberos GML2D**	**Manually Developed**
Dev. Time (Hours)	0.51 Modeling + 4.81 Implementing = 5.32 total	30.0 Implementing
Effort (LOC)	519 User Gen. + 74 XNA Gen. + 5546 Auto Gen. = 6139 total	3822 User Generated + 74 XNA Generated = 3896 total

Discussion

From Table 5.3 and Table 5.4, we can see that the amount of savings varies greatly depending on the complexity of the game. We attribute the difference in the effort percentage between Pong 29% and SpaceKatz 86.4% to the expressiveness of our language, and the level of complexity of the games directly. The game of Pong consists mostly of interactions. As a result of this, while we were able to model a large set of the user input game representation, there was also a bit of game behavior that could not be modeled with the current version of **Eberos GML2D**. In comparison, with a more complex game like SpaceKatz, which requires a lot of implementation of user and game logic, as well as enemies' AI, we obtained a better savings of 86.4% less lines of code. With the logic section of **Eberos GML2D**, we were capable of modeling a large section of the agent logic for the enemies, menus, and player, something that had to be otherwise implemented without using our approach. Similarly, we were able

to model the entities, bullets, asteroids, and pool of entities using **Eberos GML2D**, which had to be implemented for non-DSL version of the game.

4.3 Knowledge Base - Expert System

In this section we present the scenarios provided by our expert and from them we select some of the original identifier which will be utilized by our expert system to aid in the image analysis.

In the medical domain of hair restoration the assessment of a hair transplant candidates begins by looking at the area of thinning or balding and assessing that surface area that will be transplanted in cm^2 . After the assessment is performed, the area from which the hair will be taken (referred to as the donor area) is assessed by looking at its size (that is its height and horizontal length as seen on Figure 4.3). The surgeon also grossly looks at the **visual density** and performs a visual assessment of **hair diameter**. As seen in slide three magnified photographs of shaved areas measuring 0.5 cm^2 are taken and assessed. Then the percentages of **one**, **two**, **three**, and **four hair follicles** in each **follicular unit** is determined as well as the **number of follicular units** are clusters per cm^2. Hair diameter or caliber ranges from 50μ which is very low to 70μ which is a good, coarse hair diameter.

Worst Case Scenario: In a worst case scenario a patient has a baldness such as a type VI or VII as seen on the Norwood scale in Figure 4.4 and the donor density is 50 follicular units per square centimeter (very low) and a high proportion of **ones** vs. **twos** or **threes** hair follicular units. This patient would have to accept a very small

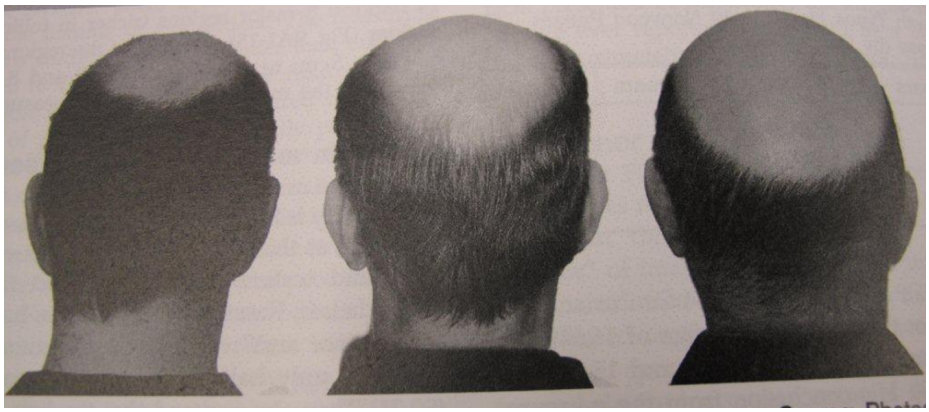


Figure 4.3: Donor Areas in three different patients.

frontal area of transplantation or may in fact be rejected for the procedure especially if, aside of his low density the hair diameter is very low.

Figure 4.4: Norwood Scale - Baldness VI - VII

Best Case Scenario: In a best case scenario a patient as a baldness type III male pattern as seen in the Norwood scale in Figure 4.5. In this patient these areas could be filled even with a low density of 60 follicular units per cm^2. If a patient with a type III baldness has high donor density (80 follicular units per square centimeter) then he could not only fill completely the areas of need, but could have a second procedure to increase density. As In the first procedure the patient would need to have the follicles implanted with some distance between them (i.e. no higher than 35 follicular units per square centimeter), so that there is sufficient blood flow to the grafts to achieve good survival and growth.

Figure 4.5: Norwood Scale - Baldness III

Based on the provided scenarios we have been able to determine the following indicators that our expert system can utilize to help the physician:

- Proportions follicular units composed of one, two or three follicles
- Donor density
- Hair density
- Hair diameter

Based on these and the scenarios presented above we can produce some of the rules that can be used by the proposed expert system. Rule 4.1 was extracted from the worst case scenario and we will refer to this one as **Rule #1 - Patient Rejection**

$$\begin{aligned} &IF\ proportionOf(ONES, TWOS)\ IS\ HIGH \\ &\ AND\ hairDiameter \leq 50\ \mu \\ &\ THEN\ Reject\ Patient \end{aligned} \tag{4.1}$$

Similarly we can extract some rules from the best case scenario we can extract some rules to suggest the physician possible procedure based on the current information on

the patient. Such is the case of rule 4.2 which we will refer to as **Rule #2 - Density Increase Recommendation**.

$$IF\ donorDensity\ IS\ HIGH \\ THEN\ RecommendFor(HighDensityImprovent) \tag{4.2}$$

As we continue to explore more scenarios with our collaborators in the medical field we will generate a set of criteria to be used for generating the rules for our proposed expert system. This scenario will provide us with the information that will allow us to determine ranges such as **HIGH** and **LOW** for each of the indicators that we have extracted from the previous scenarios.

4.4 Generic Mobile Framework

In this section we discuss our preliminary work on the generic mobile framework. Subsection 4.4.1 discusses the high level design of the framework and describes its components. Subsection 4.4.2 discusses the current state of the framework. This subsection also discusses the principles and methods used during the implementation for the current framework.

4.4.1 High Level Architecture

During the original discussions with our collaborator in the medical field we isolated a set of components which must be available for the creation of medical apps. The

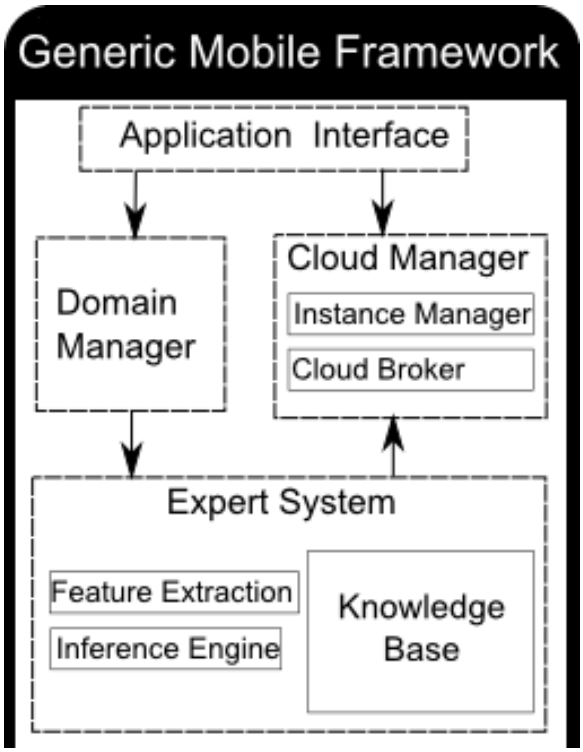


Figure 4.6: High Level Architecture of the Proposed Mobile Framework

list was then revised based on our expertise and we arrived at a set of components we believe encompass the needs of most mobile medical application. The list of components we arrived at is:

- Application Interface
- Domain Manager
- Cloud Manager
- Expert System

Based on this components we then developed the high level architecture shown in Figure 4.6

Application Interface: The application interface is the facade presented to the applications developed on top of our proposed framework. This layer abstracts all the functionality of our framework and makes it easily accessible to developers. This also serves as a divider between the user apps and the rest of our architecture thus easily allowing for the future modification of our framework with little effect preexisting applications. This can be seen as the broker between application code and our framework code. This components delegates all the application calls to the **domain manager** and the **cloud manager**.

Domain Manager: The domain manager controls what systems should handle the application calls received from the **application interface**. The response of the domain manager can vary depending on the specific medical domain that the external application is trying to address. Based on the nature of the application the domain manager perform any of the following operations:

- Replace feature extraction framework
- Replace the inference engine
- Replace the knowledge base
- Replace the existing expert system in its entirety

Since the domain manager seats between the application interface and the rest of the app, it allows for new functionality to be added to the framework and then exposed to the application interface without affecting the user applications.

Cloud Manager: In some cases, some images may require processing power beyond that available in the device. This means our framework must find some ways

to still provide a satisfactory solution. In order to compensate for this our framework provides the cloud manager which can offload some or all of these operations to the cloud. In order to allow for this the cloud manager provides an instance manager and a cloud broker.

- Cloud Broker: Given the large number of cloud providers for public and private there is a need to have a component to handle the tasks of accessing, registering and removing cloud providers. The cloud broker allows for ease of interfacing with these provide without requiring any additional interaction for the external application. The cloud broker can either require the external application interface to provide a cloud provider of its choice or assign a default provider in the case that no provider is specified. It also allows for optimization performances by selecting the best or cheaper provider based on application specification.

- Instance Manager: Once a cloud provider has been selected the instance manager handles all the cloud operations required to perform the task. These operations include: instance creation, destruction, modification. It also handles the load balancing on the cloud by scaling the instances horizontally or vertically depending on the needs of the external application.

Expert System: This is where the expert system for the framework is located. The design should allow for the expert system or parts of to be replaced at runtime pending on request of the domain manager. In the case of this expert system the user interface has been replaced by the domain manager which forwards the queries and other requests to the expert systems. Much like other expert systems its has a knowledge base, an interface engine and a feature extraction component.

- Feature Extraction: This component uses the labeling framework proposed by this work to analyze medical image data and provides the expert system with information needed to process any query requested by the domain manager.

- Inference Engine: The **inference engine** in an expert system uses the rules and facts stored in the knowledge base to validate assertions or provide suggestions to the user. The extent to which the inference engine can provide accurate assertions depends greatly on the type of expert system. Common types of expert systems include: fact-based expert systems, frame-based expert systems and semantic network expert systems. In the case of rule-based expert systems the success rate is directly proportional to the completeness of the knowledge base and can provide results that rival experts but cannot deal too well with uncertainty. On the other hand network and fuzzy-based expert systems tend to have a lower success rate but can deal especially well with uncertainty since the expert knowledge is expressed to allow more flexibility.

- Knowledge Base: The knowledge base is where the knowledge of the expert system is stored. This knowledge is usually gathered by the knowledge engineered and added for interpretation by the inference system. The most common method for representing the expert knowledge is in the form of rules. The knowledge base may also contain facts about the domain. These facts can be descriptions of objects, as well as relationships between the objects. Using these facts and rules in the knowledge base the expert system can then proceed to determine whether assertions about objects are true or false.

4.4.2 Current Architecture

So far we have managed to develop an initial framework which has been successfully used to develop a medical mobile application for the hair transplant domain. In this subsection we continue to describe our preliminary work by explaining the components of the currently implemented framework.

Application Interface

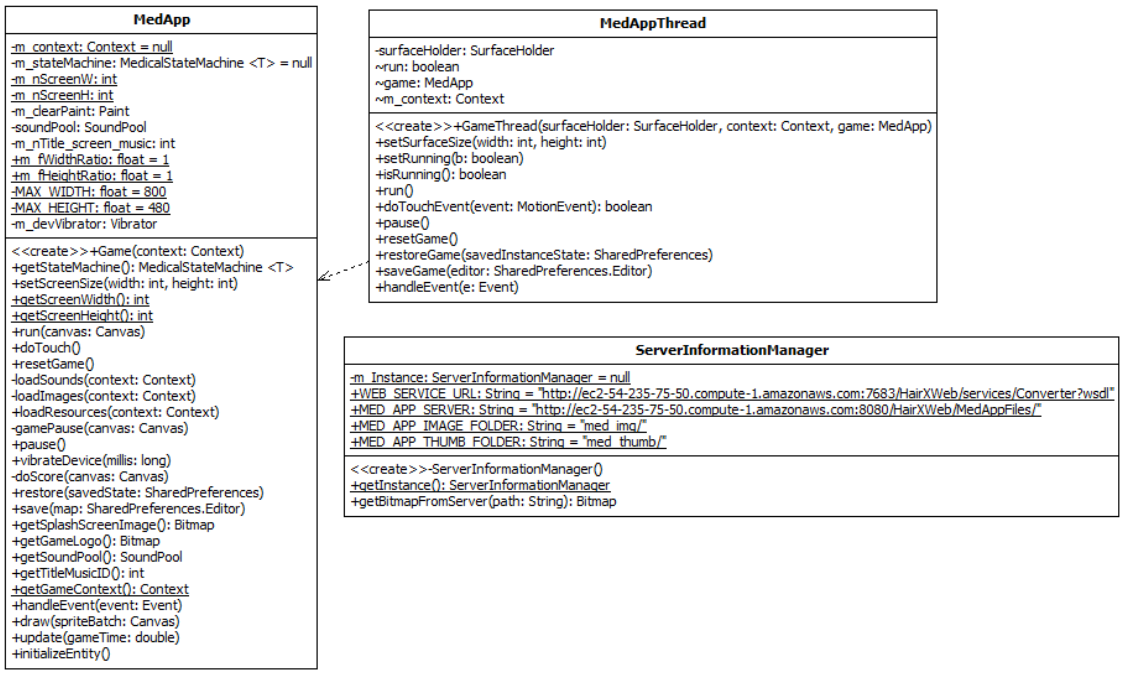


Figure 4.7: Class Diagram for the Application Interface

Figure 4.7 shows the current structure of the application interface mentioned in Section 4.4.1. This component is composed of the following classes:

- **MedApp:** This is the main class that the applications that are to be built using our framework must extend. It is the main entry into our framework and contains all the functionality needed to create a simple mobile application.

- **MedAppThread:** This is the class that controls most of the flow of MedApp. This class is an internal class and not exposed to the user. This class drives the implemented user class.

- **ServerInformationManager:** This class handles the interaction with the Cloud Manager. This separates the Application interface from the Cloud Manager using the Bridge design pattern which allows us to replace the entire implementation of the class at runtime if needed. This not only allows for the replacement of the Cloud Manager but also allows us to switch between Cloud Managers at runtime if there is a need to choose a more optimal one.

Expert System Broker

Figure 4.8 shows the current structure of the expert system broker mentioned in Section 4.4.1. This component is composed of the following classes:

- **MarkerExpertInfoActivity:** This class handles the application view for displaying the expert system information to the user. This class receives the information from the expert system's feature extraction and dispatches it to the application interface to be handled by the user application. This dispatch is performed by the application event notifier systems which all applications must register with in order to receive information from the system.

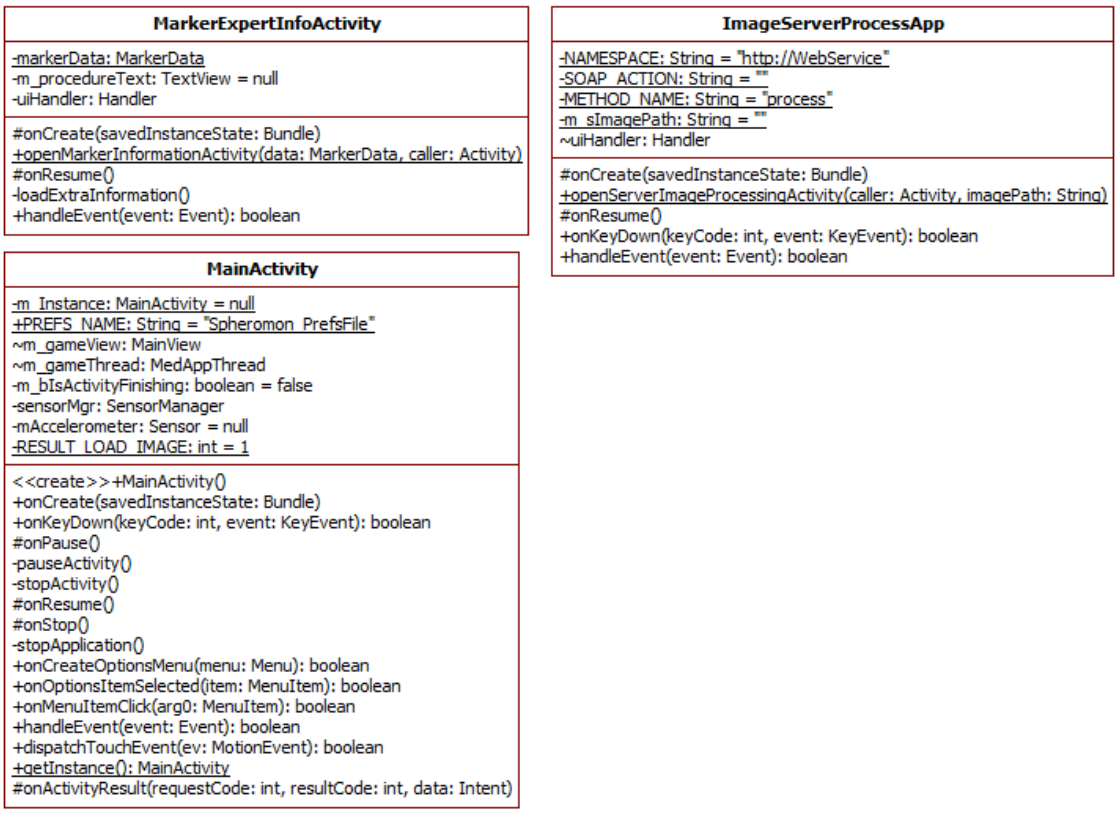


Figure 4.8: Expert System Broker

- **ImageServerProcessApp:** This class handles the interaction between the expert system and the cloud manager. This class is in charge of notifying the cloud manager when the operation requires more resources than the mobile device can supply. In the event that no resources are available locally this class forward all the processing to the cloud and awaits the response from the system to so the expert system can continue processing the query and produce any recommendations.

- **MainActivity:** This represents the basic class that handles all Android dependent functionality in the framework.

Feature Extraction

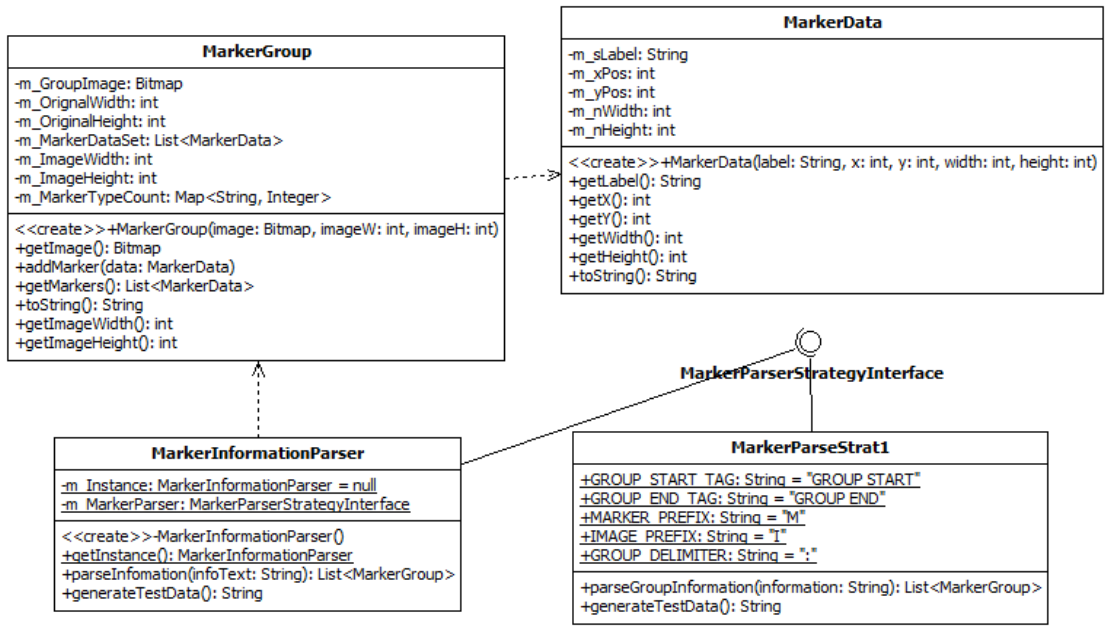


Figure 4.9: Class Diagram for the Feature Extraction

Figure 4.9 shows the current structure of the feature extraction component mentioned in Section 4.4.1. This component is composed of the following classes:

- **MarkerData:** The MarkerData class holds the information for a given marker returned by the feature extraction component. The marker can be extended to meet the needs of the applications being developed. At its most basic form it contains a label and the position information to place it on the analyzed image.

- **MarkerGroup:** This class represents a cluster of MakerData items. This class is used to represent the markers obtained from the feature extraction component as a whole. This grouping is specially relevant when dealing with collaboration among doctors. Each group will have a color identifier for every doctor or nurse to enhance collaboration.

- **MarkerInformationParser:** This class handles the interpretation of the information produced by the feature extraction component and translates it into MarkerData and MarkerGroup objects, respectively. This class uses the Strategy design pattern to allow for the parser algorithm to be replaced and allows for different kinds of data representation by the expert system. Since different systems can have their own proprietary way of representing information, there is a need to provide means to replace the parser without having to modify the architecture of the system.

- **MarkerParserStrategyInterface:** This interface is created as part of the Strategy design pattern which allows for the system to be able to change parsing algorithms at runtime without modification of the architecture.

- **MarkerParser:** This class handles the parsing of the data obtained from the feature extraction component. In order to allow for flexibility and different types of parsing algorithms by delegating the parsing to the MarkerParserStrategy.

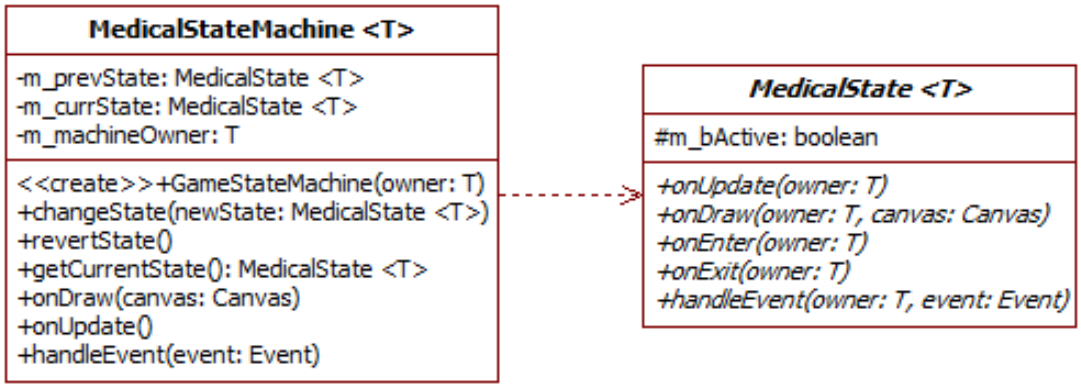


Figure 4.10: Class Diagram for the State Manager

State Manager

Figure 4.10 shows the current structure of the state manage component. The state manager component allows for developers to create functionality on their mobile application that is controlled by a finite state machine. This component is composed of the following classes:

- **MedicalState:** The MedicalState class is part of the implementation of the State design pattern and is used to allow for the implementation of state-based operations. Each state that needs to be implemented can be done as an extension of MedicalState and given to the MedicalStateMachine object for management.

- **MedicalStateMachine:** This class is also implemented as part of the State design pattern. This class specifically simulates the functionality of a finite state machine (FSM). For every class in the external apps that wants to implement a FSM controlled operation all that is required is for an instance of this class to be referenced.

Entity Event System

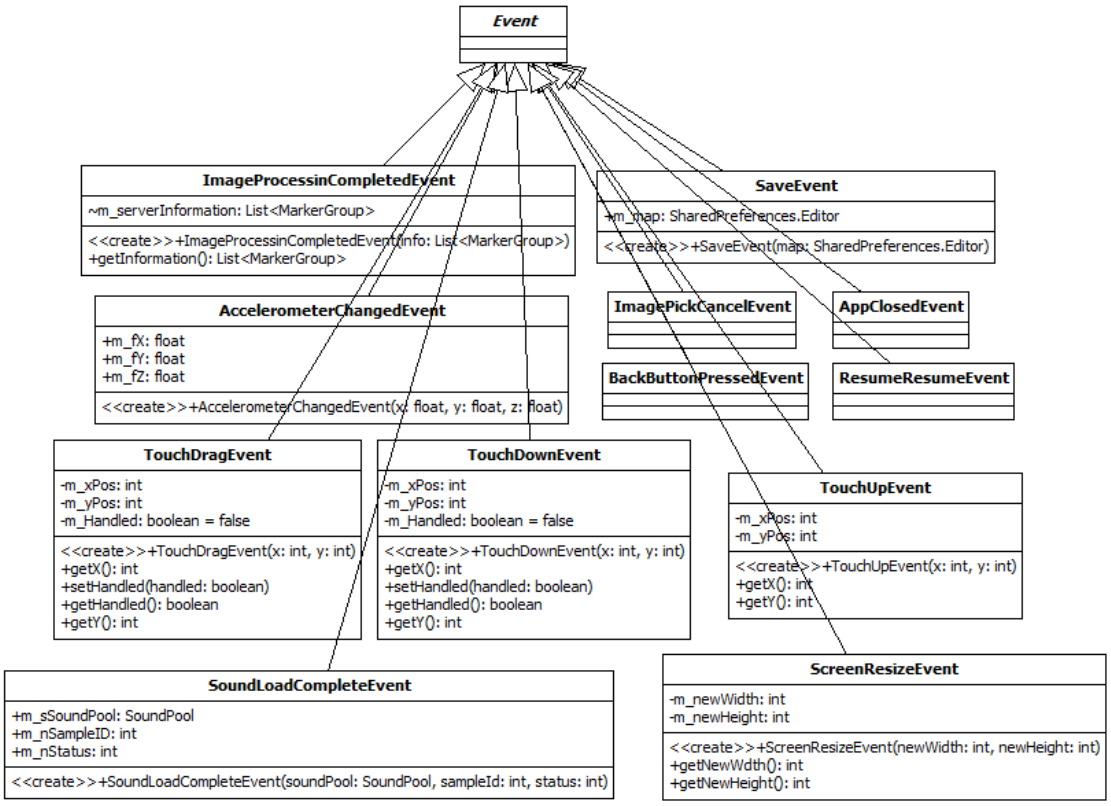


Figure 4.11: Class Diagram for the Entity Event System

Figure 4.11 shows the current structure of the entity event system. The entity event system allows for a reduction in the dependency of mobile applications by allowing entity objects to send information to one another without being dependent on their interface. This system is implemented taking advantage of the Observer design pattern to developed the subscriber-consumer relationship needed to reduce the a fore mentioned dependency. This component is composed of the following classes:

- **Event:** In allowed for a decoupled mean of passing information between the components in the framework and mobile application built on it we integrate

an event management system. The event class represent the parent of all events that can be sent in the system. Any new events that would be required by user applications can be added to the framework by simply extending this class.

- **ImageProcessingCompletedEvent:** This event notifies that the processing of the image is complete. This is usually signaled when the feature extraction component is done processing a given image. This event also contains the information about the features extracted from the image. This information usually comes as a list of marker groups.

- **SaveEvent:** This event is used to signal the request to persistently store information on the device. This could be information about the user preference or any updates to the expert system that must be available when the application starts up again or the next time the users opens it.

- **AccelerometerChangedEvent:** This event is used to signal the change of the values read from the accelerometer in the device.

- **ImagePickedEvent:** This event notifies the framework that an image has been selected and is ready for processing.

- **AppClosedEvent:** This event is native the the mobile platform and wraps around the notification that the application has been closed to by the user. This is wrap so that it can notify the applications built using our framework of this event so they can perform any clean up that is needed.

- **BackButtonPressedEvent:** This event is native to the the mobile platform and wraps around the notification that the application has been paused. An application in a mobile device may be paused when a phone call has been received so applications developed on mobile devices must pause its functionality until the call is over. This is wrapped so that it can notify the applications

built using our framework of this event so they can perform any clean up that is needed.

- **ResumeResumeEvent:** This event is native to the mobile platform and wraps around the notification that the application has resumed. This is wrapped so that it can notify the applications built using our framework of this event so they can perform any initialization that is needed.
- **TouchDragEvent:** This event notifies when a drag on the screen of the mobile device has been detected.
- **TouchDownEvent:** This event notifies when a touch on the screen of the mobile device is detected.
- **TouchUpEvent:** This event notifies when a touch has been released from the screen of a mobile device.
- **SoundLoadEvent:** This event notifies the system that a sound asset has been loaded onto the system memory and is ready to be used by the application.
- **ScreenResizeEvent:** This event notifies the resize of the application screen. Resizing usually occurs in mobile applications when the device orientation is changed. This can occur on some devices if the application is started while the device is locked, so some applications may need to respond to this event.

Activity Event System

Figure 4.12 shows the current structure of the activity event system. The activity event system allows for a reduction in the dependency of mobile applications by allowing activities to send information to one another without being dependent on

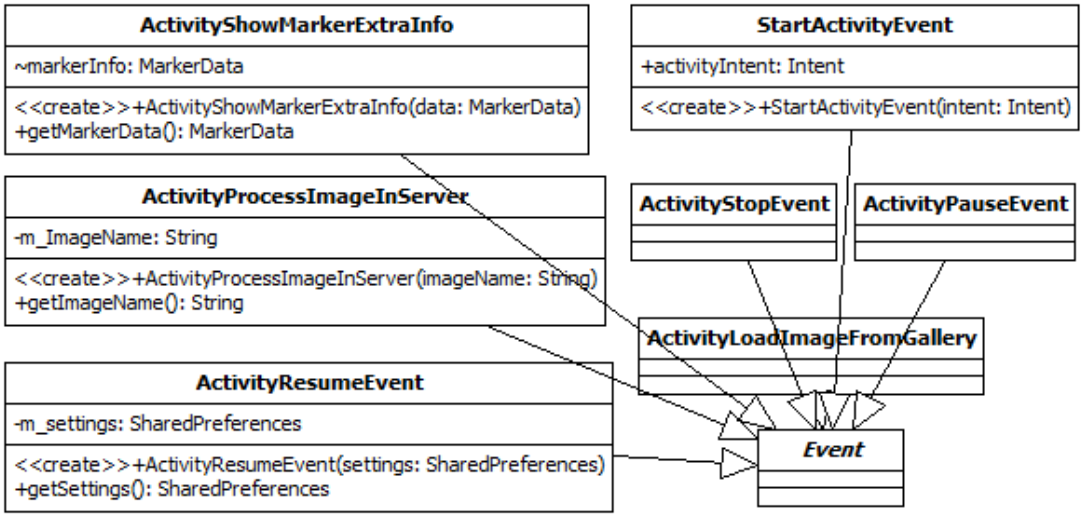


Figure 4.12: Class Diagram for the Activity Event System

their interface. This system is implemented taking advantage of the Observer design pattern to developed the subscriber-consumer relationship needed to reduce the a fore mentioned dependency. This component is composed of the following classes:

- **ActivityShowMarkerExtraInfo:** This event notifies activities to show extra information regarding the contained markers. Each activity is free to display this information in any way they choose. This can be a pie chart, a bar chart, or any visual representation.
- **ActivityProcessImageInServer:** This event notifies an activity to try to process the given image on the cloud server. Activities that don't handle this can simple ignore the event.
- **ActivityResumeEvent:** This event wraps around the platform's resume notification. It allows the framework to notify existing activities to resume operations.
- **StartActivityEvent:** This event notifies the request to start.

- **ActivityStopEvent:** This event wraps around the platform's stop notification. It allows the framework to notify existing activities to stop operations.
- **ActivityPauseEvent:** This event wraps around the platform's pause notification. It allows the framework to notify existing activities to pause operations.
- **ActivityLoadImageFromGallery:** This event notifies the framework to load the platform's gallery component so that the user is able to select a picture from the device's gallery.

Event Dispatcher System

Figure 4.13 shows the current structure of the event dispatcher system. This manages the dispatching of event between objects as well as between activities. This component is composed of the following classes:

- **MedicalActivity:** This class provides an abstraction to allow for the implementation of activities on medical mobile application. An activity must be a MedicalActivity if it is to be notified by the MedicalActivityNotifier.
- **MedicalActivityNotifier:** This class allows for medical activities to be register so they respond to events. This class implements the Observer design pattern to allow for the support of a subscriber-producer methodology.
- **Entity:** Entity represents any other objects that are not activities. If an object inside a medical application is to react to events it must be an entity type and extend this abstract class.

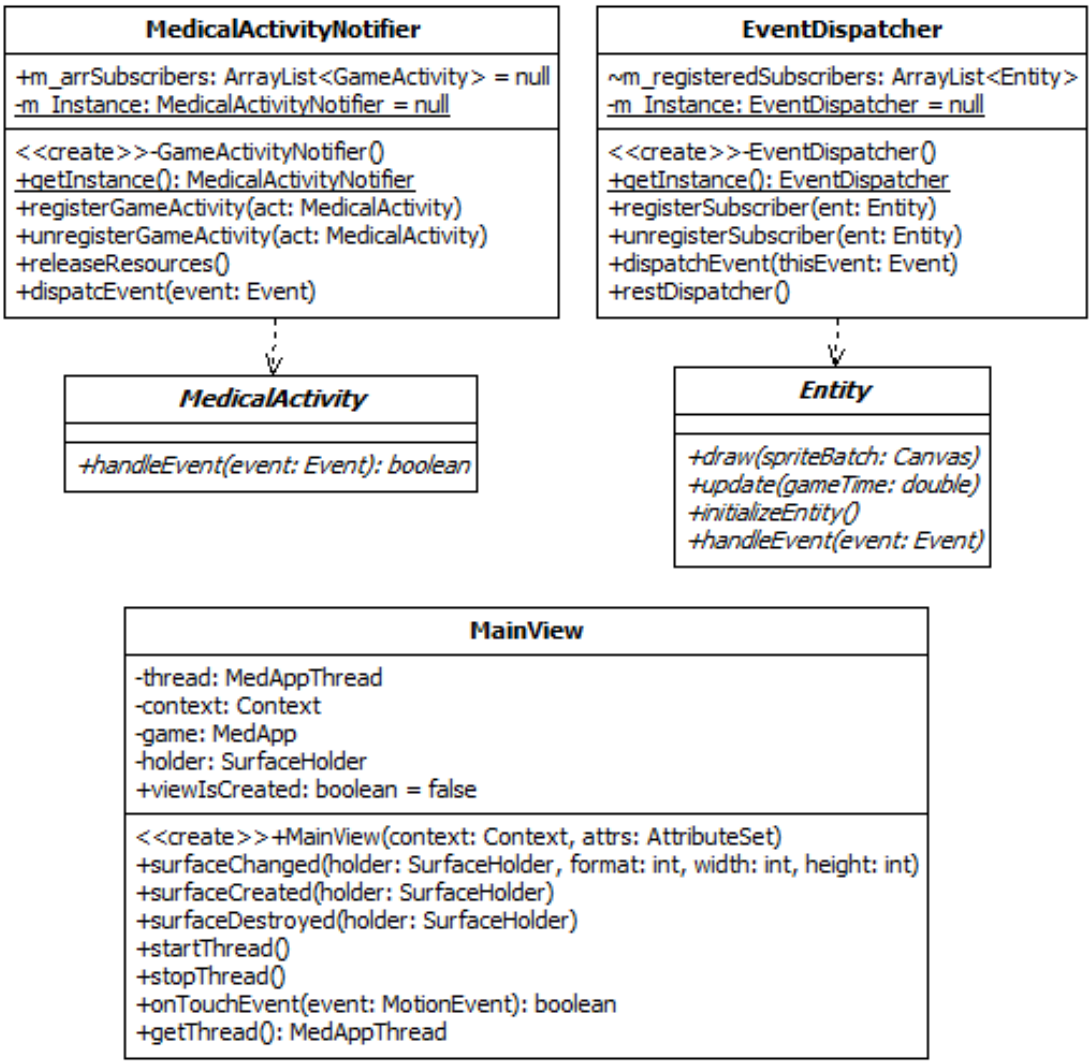


Figure 4.13: Class Diagram for the Event Dispatcher System

- **EventDispatcher:** The event dispatcher notifies all entities registered within it of events that take place inside the framework. This also allows for entities to communicate among each other with events created by the user. This class implements the Observer design pattern to support this subscriber-producer methodology.

- **MainView:** This class represents the view where extra visual information such as pie chart and other graphical information representation means can be drawn.

Patient Record Management Module (PRM)

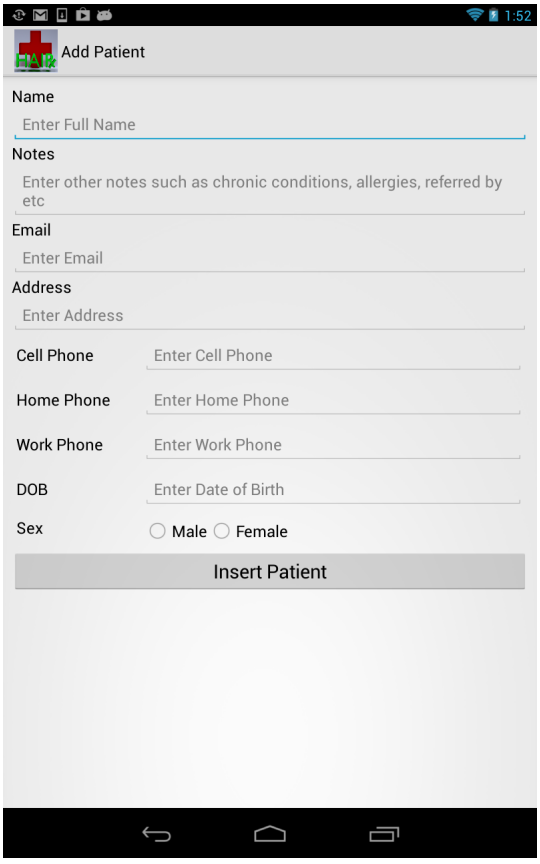


Figure 4.14: Base Add Patients View

This module provides developers with a base implementation for handling patient records. The framework currently provides developers with means to store new patients into their mobile medical application. The same way it provides developers with a way to update the patient records as well delete patient records from the application. Currently this provides a set of patient fields common to many medical applications we explored. These fields include: patient name, patient notes, email, address, cell, work and phone number as well as sex and date of birth. This information is contained inside the patient model which can be extended by the developer to add additional fields if the application requires it.

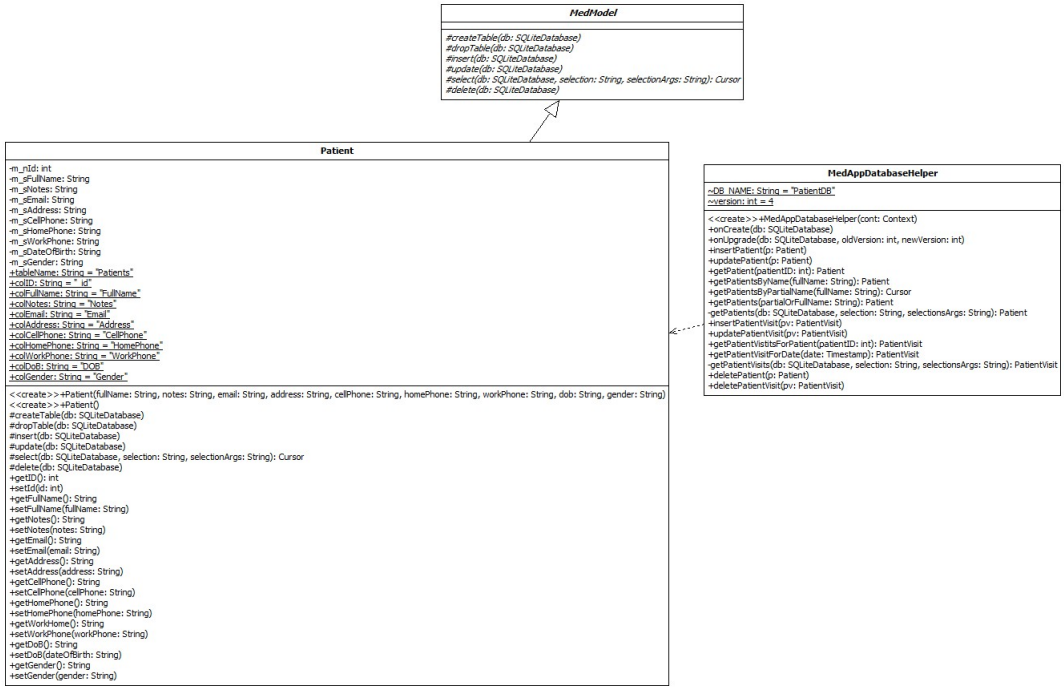


Figure 4.15: Patient Record Model - View

In order to provide developers with the utmost flexibility we have implemented our framework following the Model-View-Controller paradigm. This allows developers to create new modules no existing in our framework as well as extends the already existing modules to match their needs. In the patient record module we have created the following classes to handle patient record data.

- **MedModel:** This class represents the model in our framework. It is an abstract class which contains the minimum required methods in order to new models to be able to incorporate into our framework. This is the class that developers must extends if they intend to add any additional models to represent data in their applications.

- **Patient:** This class represents the patient data in our PRM. It encapsulates all the fields needed to handle a patient record. Each model is aware of the steps to retrieve records from the database as well as store records. Any encryption and decryption of records is also handled by this class.

- **MedAppDatabaseHelper:** This class wraps around the database for easy information retrieval. It is implemented following the repository pattern and hides all implementation details needed to deal with the database from the users.

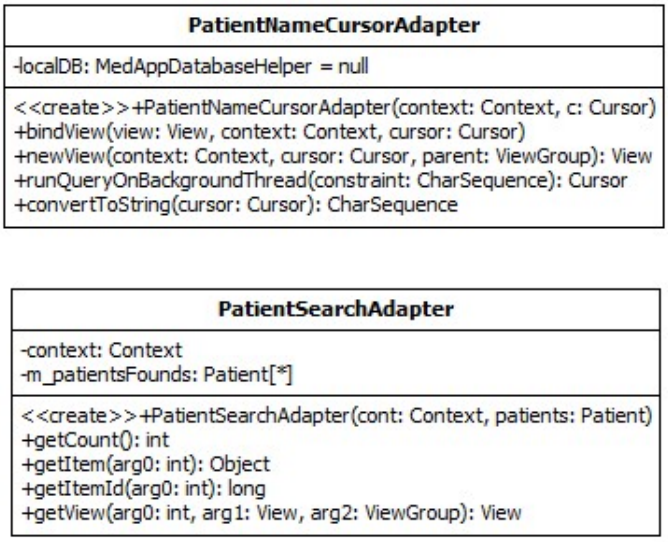


Figure 4.16: Patient Record Module - Controller

Controllers handle any logic and operations that might need to be perform. In the case patient records this might include retrieval of a specific patient, or sorting the patients based on their last appointment date.

- **PatientNameCursorAdapter:** This class handles the auto-complete of patient names. This is handled during the search method to provide users with a quick access to patients when looking for a specific patient by name.
- **PatientSearchAdapter:** This class provides the functionality that handles patient records as a whole. This also deals with providing the view with a list of patient records to display.

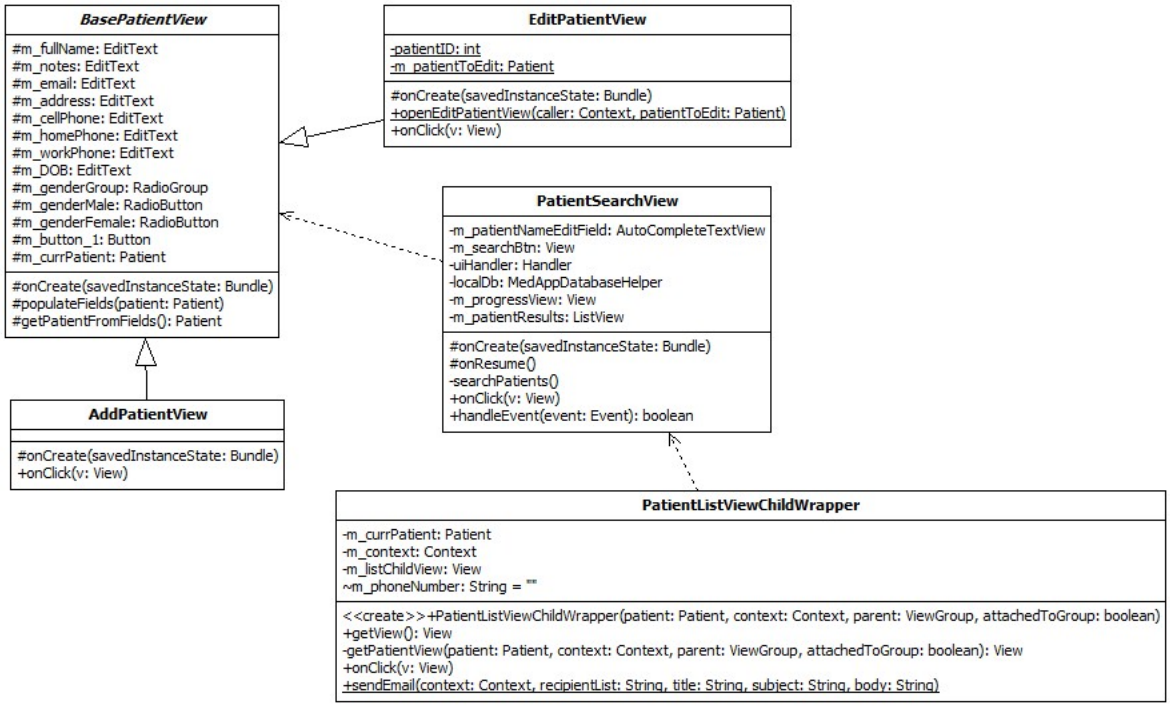


Figure 4.17: Patient Record Module - View

The view component of the patient record module (PRM) deals with all the functionality required to display the data to the user. This also handles how the data is organized on to the device screen.

- **BasePatientView:** This abstract class represents the most common components in a patient view. This class can be extended to create other views that deal with patient data. This class also handles all the functionality required to display patient information to the user interface. In the same way, this class also handles the functionality of retrieving information input in the user interface.

- **AddPatientView:** The AddPatientView class is an extension of the BasePatientView class and provides with a simple view that handles all the basic information needed to insert a patient into the records. This allows developers to reduce their implementation time by simply utilizing this class instead of having to implement one from scratch.

- **EditPatientView:** Similar to the AddPatientView, this class provides a basic implementation of all the functionality needed to modify and update a patient's data.

- **PatientSearchView:** The PatientSearchViewClass provides an off-the-shelf patient data display implementation. This class displays a given set of patients along with their contact information and the ability to view or scheduled meetings.

- **PatientListViewChildWrapper:** This class is a helper class for the search view. It handles the management of a single patient record which are then loaded in the patient search view.

Patient Visit Management Module (PVM)

Similar to the Patient Record Management (PRM) component, the patient visit management (PVM) component provides the developer with means to rapidly include

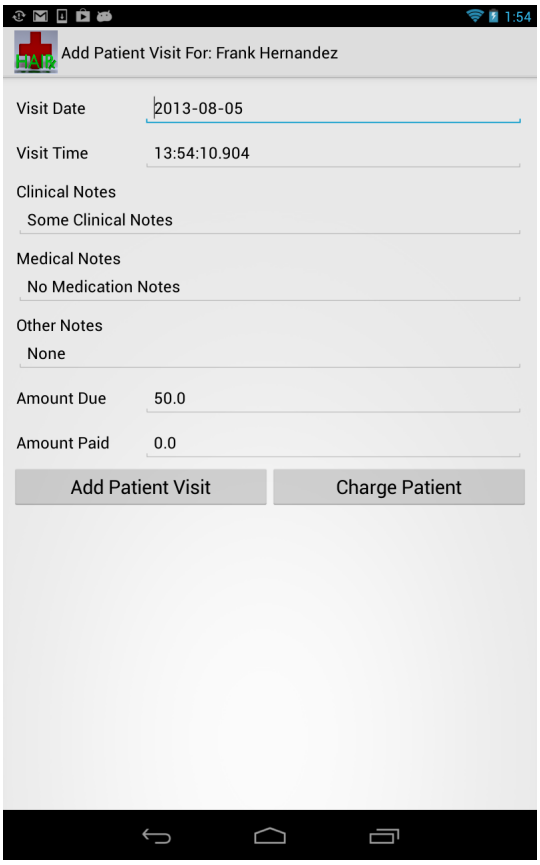


Figure 4.18: Base Add Patient Visit View

patient visit records into their mobile medical application. It follows the same Model-View-Controller architecture as the PVM which allows the developer to extend the patient visit model to include any additional information that may be required by the mobile application for the patient visits. Currently the default fields for the base patient visit are: visit date, visit time, clinical notes, medical notes, other notes, amount due, and amount paid. The patient visit management component also provides developers with synchronization of the patient visit with the doctor's Google calendar without any additional implementation.

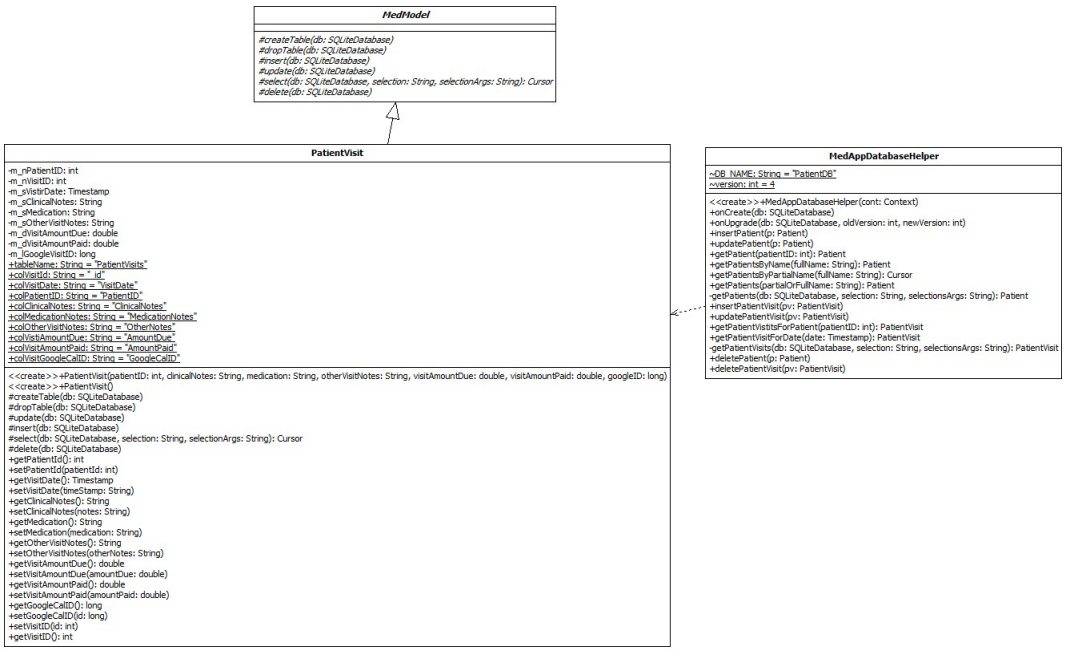


Figure 4.19: Patient Visit Model - View

In order to provide developers with the utmost flexibility we have implemented our framework following the Model-View-Controller paradigm. This allows developers to create new modules no existing in our framework as well as extends the already existing modules to match their needs. In the patient record module we have created the following classes to handle patient visit data.

- **MedModel:** This class represents the model in our framework. It is an abstract class which contains the minimum required methods in order to new models to be able to incorporate into our framework. This is the class that developers must extends if they intend to add any additional models to represent data in their applications.

- **PatientVisit:** This class represents the patient visit data in our PRM. It encapsulates all the fields needed to handle a patient record. Each model is aware of the steps to retrieve records from the database as well as store records. Any encryption and decryption of records is also handled by this class.

- **MedAppDatabaseHelper:** This class wraps around the database for easy information retrieval. It is implemented following the repository pattern and hides all implementation details needed to deal with the database from the users.

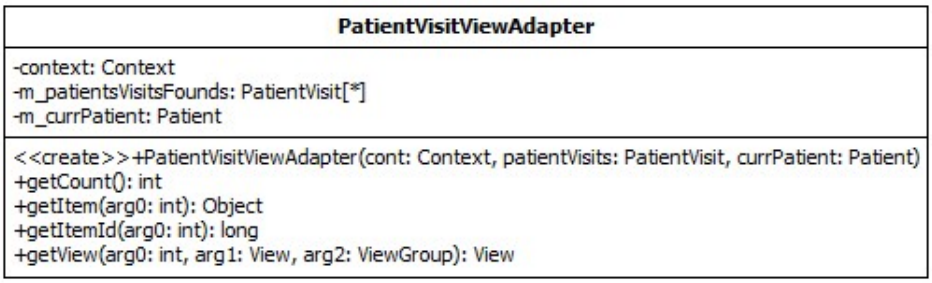


Figure 4.20: Patient Visit Module - Controller

Controllers handle any logic and operations that might need to be perform. In the case patient records this might include retrieval of a specific patient, or sorting the patients based on their last appointment date.

- **PatientVisitViewAdapter:** This class provides the functionality that handles patient records as a whole. This also deals with providing the view with a list of patient records to display.

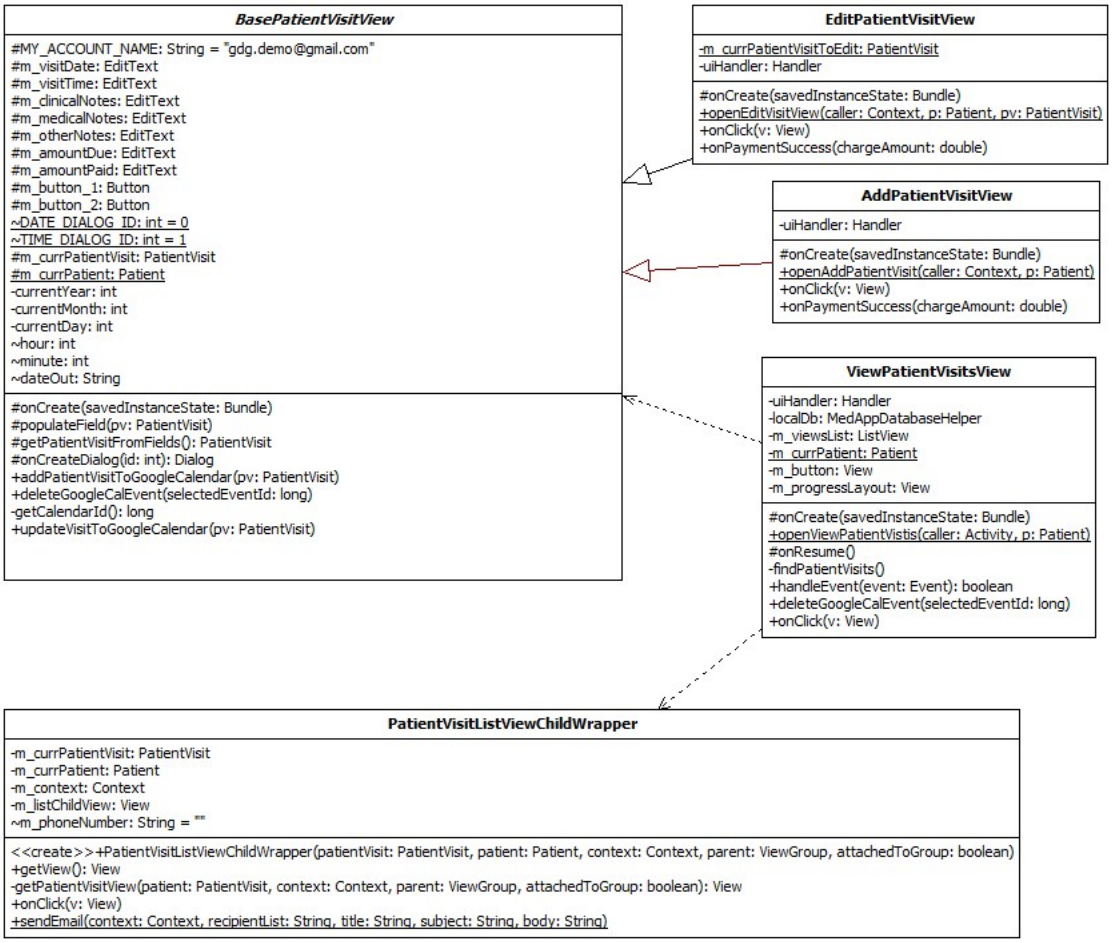


Figure 4.21: Patient Record Module - View

The view component of the patient visit module (PVM) deals with all the functionality required to display the data to the user. This also handles how the data is organized on to the device screen.

- **BasePatientVisitView:** This abstract class represents the most common components in a patient visti view. This class can be extended to create other views

that deal with patient visit data. This class also handles all the functionality required to display patient visit information to the user interface. In the same way, this class also handles the functionality of retrieving information input in the user interface.

- **AddPatientVisitView:** The AddPatientVisitView class is an extension of the BasePatientVisitView class and provides with a simple view that handles all the basic information needed to insert a patient visit into the records. This allows developers to reduce their implementation time by simply utilizing this class instead of having to implement one from scratch.

- **EditPatientVisitView:** Similar to the AddPatientVisitView, this class provides a basic implementation of all the functionality needed to modify and update a patient's visit data.

- **PatientVisitView:** The PatientVisitView class handles the implementation to display all the data for the visits of a given patient.

- **PatientVisitListViewChildWrapper:** This class is a helper class for the search view. It handles the management of a single patient visit record which are then loaded in the patient search view.

Billing Module

The billing module provides developers with a simple way to integrate credit card payments into their mobile medical application. In the current version the credit card integration is performed through a service called Stripe but can be further extended to support other billing API such as the PayPal or Amazon API. This component provides the developers with a mobile implementation of all the tools to integrate

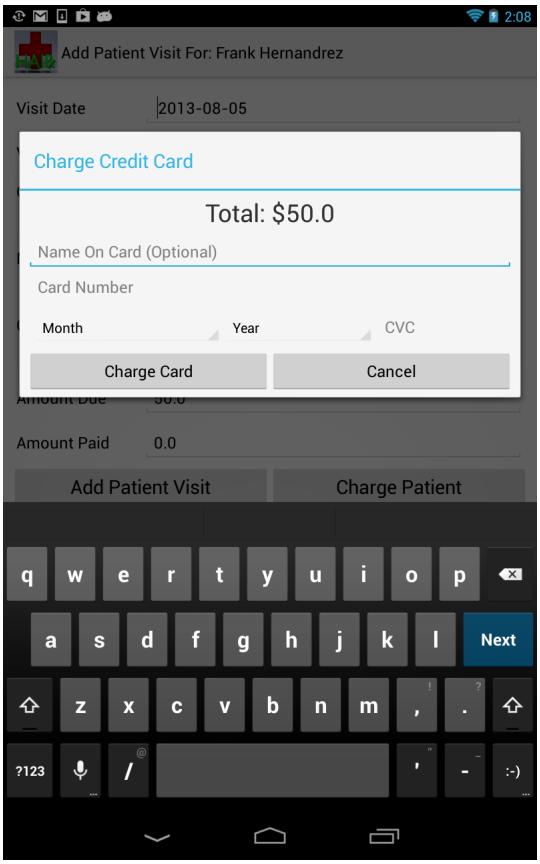


Figure 4.22: Charge Card View

payments such as authentication of the app with the services as well as the secure processing of credit cards.

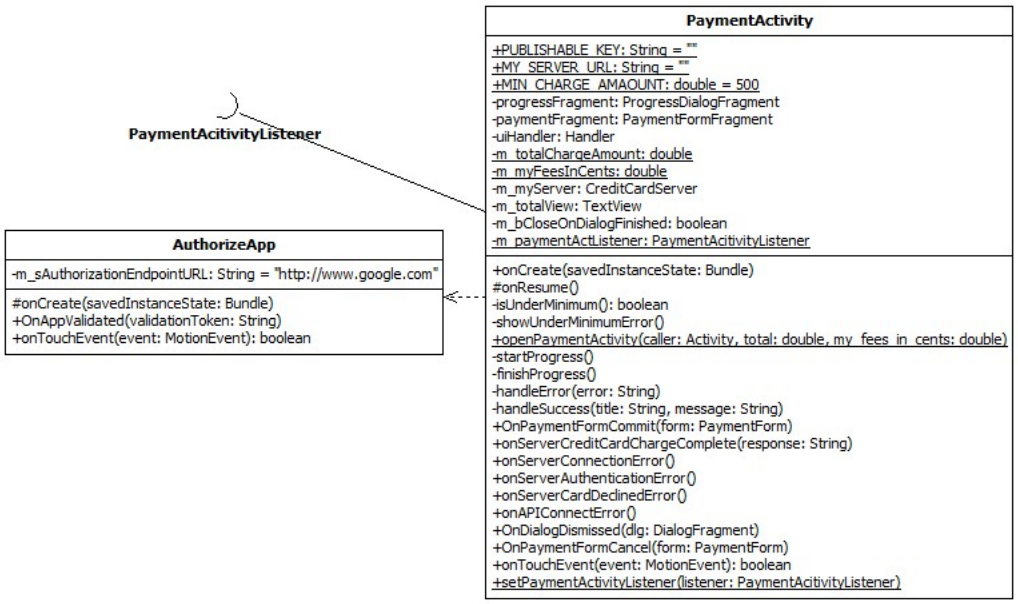


Figure 4.23: Payments Module

In order to allow developers easy include billing into their medical applications we have abstracted this functionality into two classes. It is our goal to reduce the complexity of using our modules to a minimum.

- **AuthorizeApp:**In order to use most credit card providers from within a mobile application, the application must first be authorized with the provider's server. This class provides a ready-to-use implementation that handles the authorization process for the developers. This class contacts the server for an authentication token and returns the authorization from the server if the user was a valid user. In the event of a failed authentication the developer can request the user to re-authenticate. All of this is provided as part of our billing module in order to simplify the development of medical applications further.

Since medical practitioners may sometimes handle payments such as patient visits, this is component that we found to be crucial to medical applications.

- **PaymentActivity:** The PaymentActivity is a basic implementation of an activity to handle payments. It prompts the user for all basic information required to perform a charge on a credit card. This class can also be extended to request more information if needed.

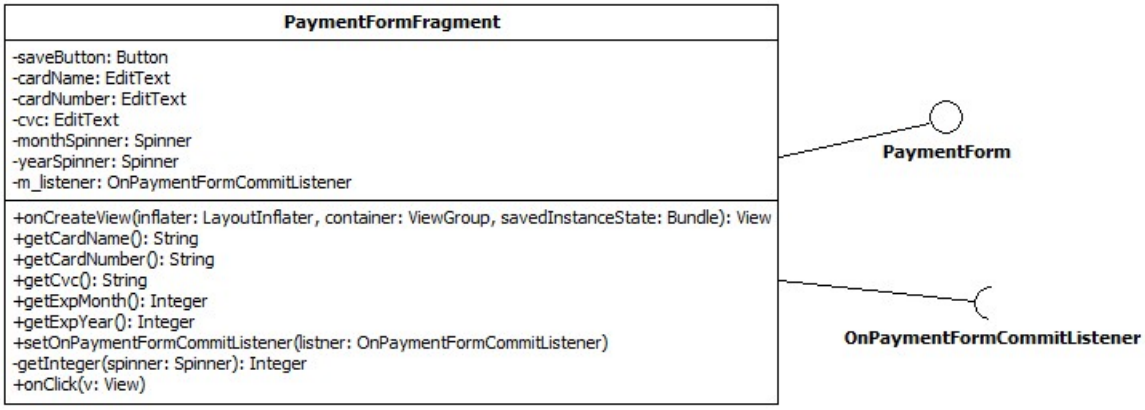


Figure 4.24: Payment Forms Subsystem

- **PaymentForm:** The PaymentForm interface represents the contract that all developers looking to implement a payment form must follow. This contains all the methods that must be implemented by any new payment form in order to properly work with our framework.

- **PaymentFormFragment:** The PaymentFormFragment is a basic implementation of a simple payment form. This prompts the user for information like the credit card number, expiration date, and CSV.

- **OnPaymentFormCommitListener:** This interface allows developers create listeners for the PaymentFormFragment. Whenever a transaction is perform in the PaymentFormFragment, it attempts to notify its listeners of this event.

This interface thus allows developer to create their own callbacks for handling payment.

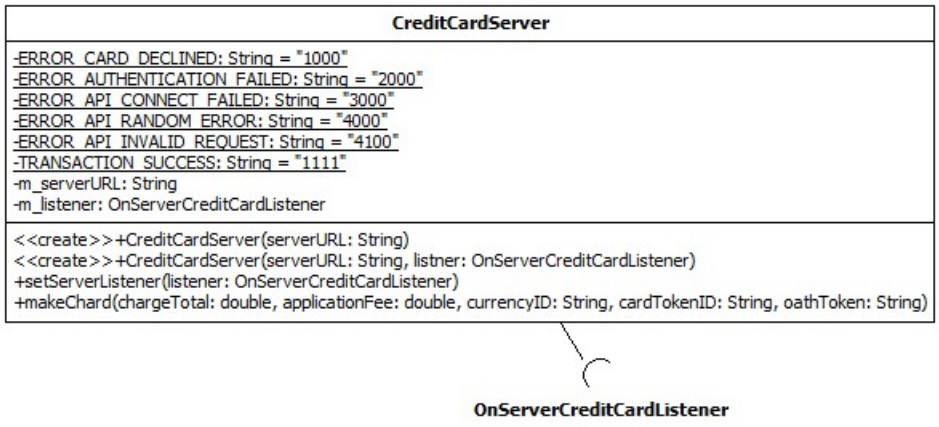


Figure 4.25: Cedit Card Handler Subsystem

- **CreditCardServer:** Once the information is obtained from the user this class handles the interaction with the credit card server. This card takes care of encrypting all the information and sending it over to be processed by the server. This class also notifies the developers of the success or failure of the transaction. Possible responses from the server include:

  1. ERROR_CARD_DECLINED
  2. ERROR_AUTHENTICATION_FAILED
  3. ERROR_API_CONNECT_FAILED
  4. ERROR_API_INVALID_REQUEST
  5. TRANSACTION_SUCCESS

- **OnServerCreditCardListener:** This listener allows developers to implement their own callback functions in order to receive the notifications from the server. This allows developers to take any action outside the ones provided by our

framework. Such transactions could include recording a failed transaction on a server of their choice or notifying the developer team when an unexpected error has occurred.

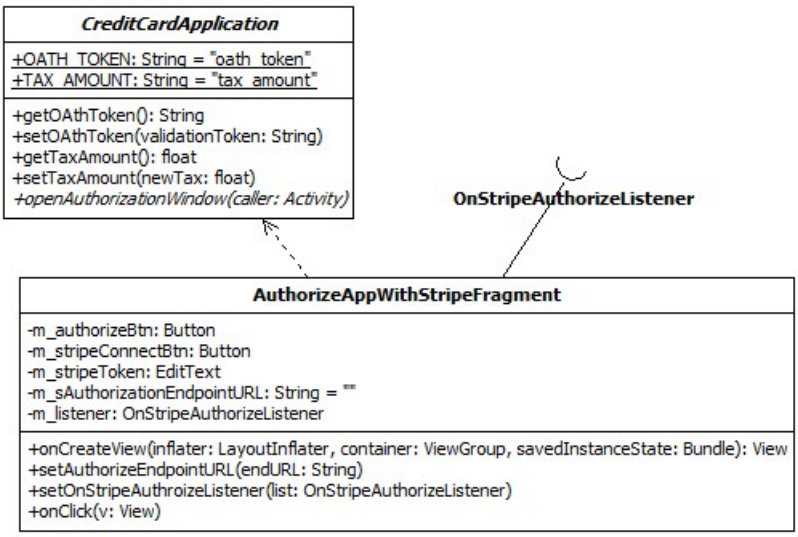


Figure 4.26: Stripe Handler Subsystem

Out of the box our framework provides developers with all the tools needed to develop mobile medical applications that handle their credit card processing through Stripe. Stripe is a service similar to PayPal but they also provide additional functionality like the ability to tack a cost onto a transaction. This has lead many developers to use Stripe over PayPal, as it allows developer to charge a fee per transactions without having to handle the transaction themselves; something that PayPal does not support. In order to accomplish this our framework includes the following classes:

- **CreditCardApplication:** The CreditCardApplication class provides the contract that developers must follow in order for their applications to be handled by our credit card processing module. It contains all the required methods that developers must implement.

- **AuthorizeAppWithStripeFragment:** This class provides an implementation of the view and models needed for the app to authenticate using the service provided by Stripe.

- **OnStripeAuthorizeListener:**This interface allows developers to implement their own callbacks which they can use to handle specific cases not handled by our framework. Whenever a user attempts to authorize with Stripe whether the transaction succeeds or fails the OnStripeAuthorizeListener is called.

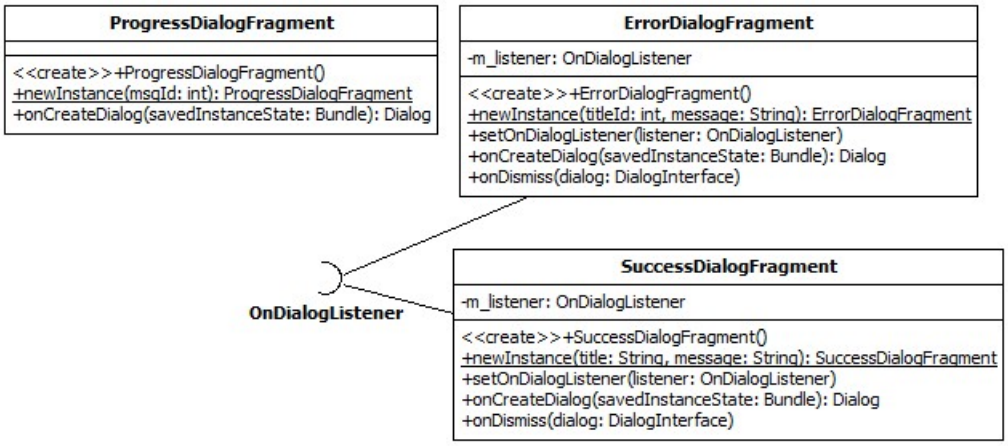


Figure 4.27: Information Display Subsystem

4.5 Case Study - HairX

In this subsection we briefly present a medical mobile application which has been built using our current version of the generic mobile framework presented in this chapter. We have codenamed this mobile application HairX and it is aimed and aiding physicians in the domain of hair transplant.

Figure 4.28: HairX - Title Screen

HairX - Title Screen

Figure 4.28 shows the title screen for our case study mobile application **HairX**. Every mobile application has an initial screen where the user begins when the application starts. This screen is usually referred to as a title screen. In order for our framework to fully support the creation of any sort of medical mobile application it had to support at least this operation. In the case of our HairX this screen was implemented using the finite state machine functionality made available by the framework. On this application every screen is represented as a state in the state machine and is handled by a global finite state machine. In the case of our case study mobile application the title screen allows the user to perform one of three operations: 1) visit the getting started screen, 2) visit the procedure screen, and 3) visit the demo app screen.

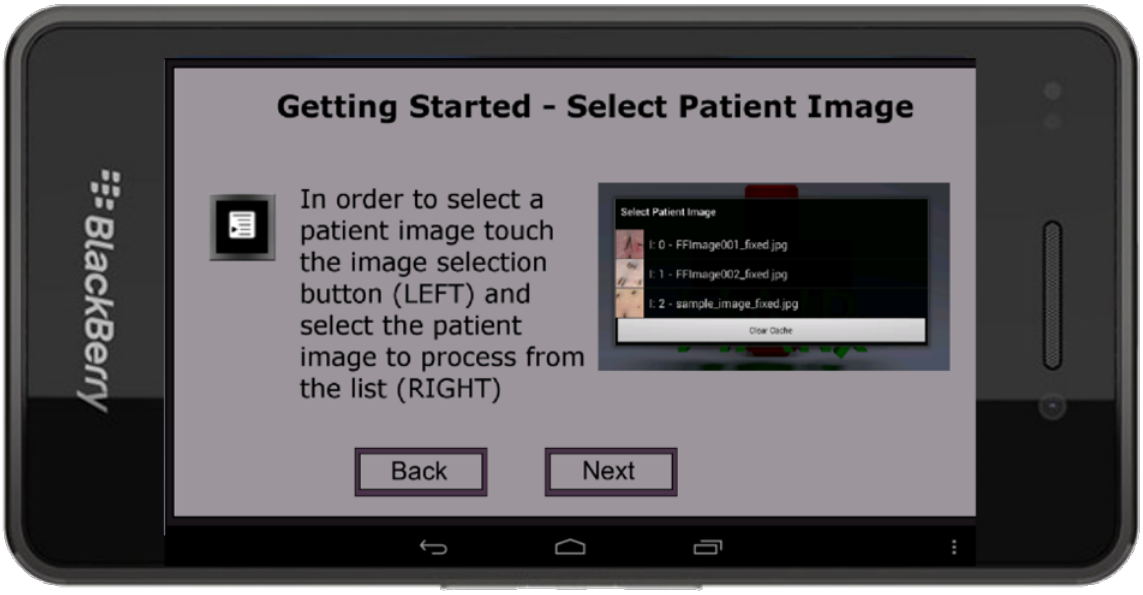


Figure 4.29: HairX - Getting Started Screen

HairX - Getting Started

Figure 4.29 shows the **Getting Started** screen. Most mobile applications which require some training to be used have a getting started section. In this section they walk the user through the functionality of the mobile application in order to quickly train him/her. HairX is no different from these applications. Our getting started screen walks the user through the different tools available to users inside the mobile application. This allows the physician or nurse to quickly get up to speed on the usage of our mobile application.

HairX - Medical Procedure

Part of the requirements for our medical mobile application was to be able to educate patients on the procedure to be performed. In HairX this is done through the procedure screen, as shown in Figure 4.30. In this screen the doctor can walk the patient

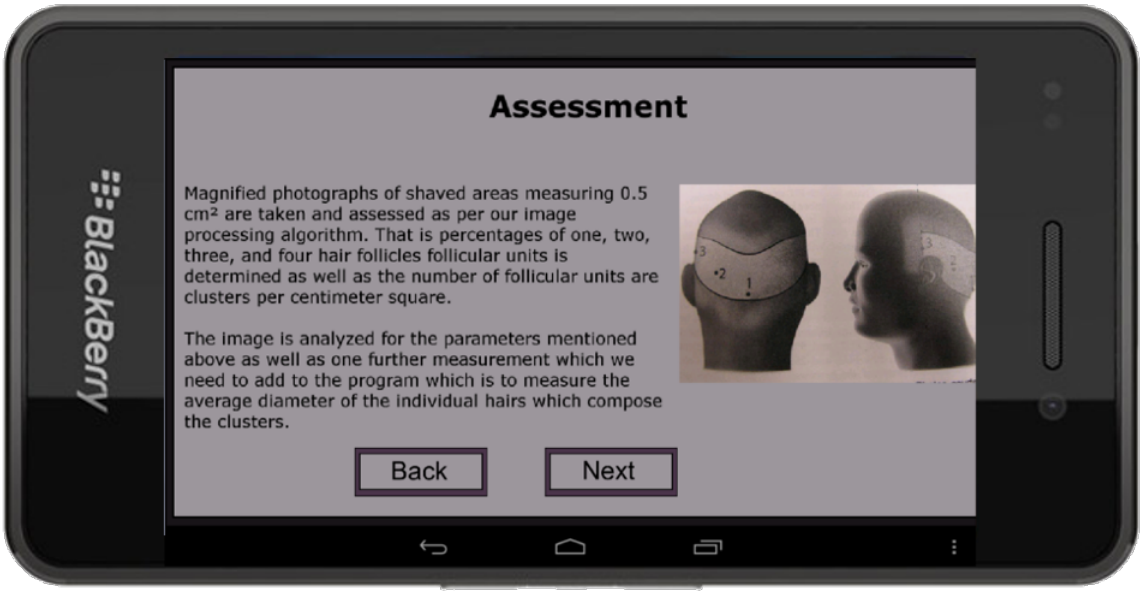


Figure 4.30: HairX - Procedure Screen

through the different steps in the procedure that the patient is about to undergo so get himself/herself familiarize.

HairX - Results

Figure 4.31 shows the application screen. In this specific screenshot, the results of our feature extraction component is shown. As required from our collaborators it detects the features from the image as follicular units. It goes further and labels the follicular units in terms of ones, twos, and threes follicular groups. Using the extracted features other important information about the patient can be obtained such as the percentage of each type of follicular group. Additionally, this step can also provide the physician with information about the individual unity such as the hair diameter. This is a step until this point was handled by close examination from a physician. HairX automates this task by sending the image through our proposed image extraction component. It is also important to point out that this stage each

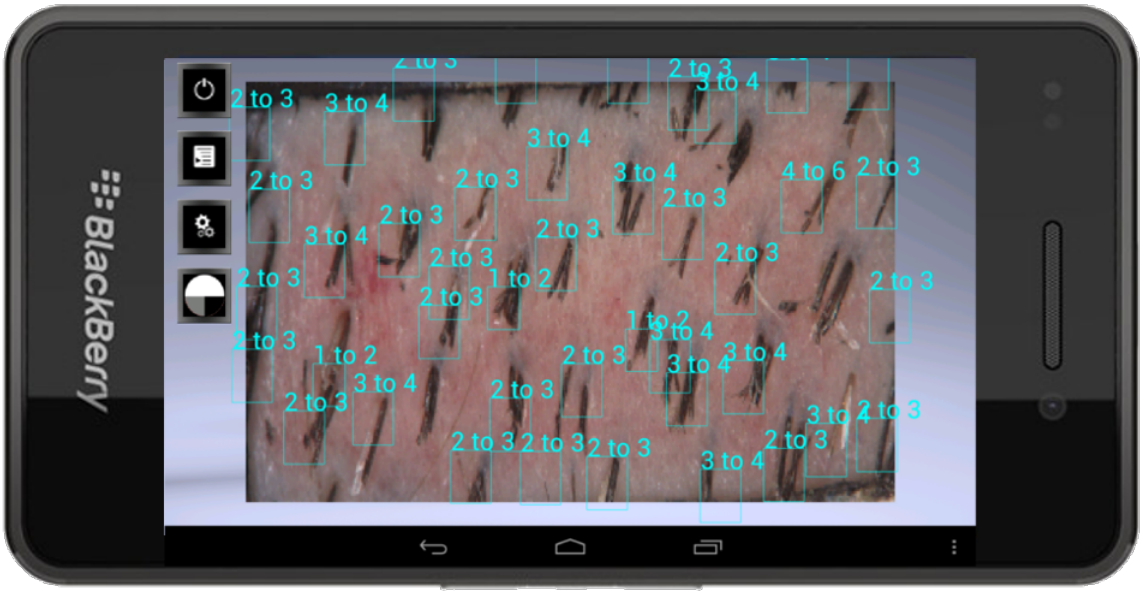


Figure 4.31: HairX - Results Screen

marker on the mobile device is touch sensitive allowing the physician to query the expert system for additional information or suggestion about common procedures to perform depending on the patient's follicle group distribution and hair density.

HairX - Extra Information Screen

Figure 4.32 shows a breakdown of the number of ones, twos, and threes follicular groups in the given image. This information is quite critical as seen from our scenarios in Section 4.3. We currently extract this information from the data provided by our feature extraction component.

Figure 4.32: HairX - Ratio of follicular Units

4.6 Case Study - GlaucoMed

In this subsection we briefly present a medical mobile application which has been built using our current version of the generic mobile framework presented in this chapter. We have codenamed this mobile application GloacoMed and it is aimed at providing physicians in areas of low resources with means to diagnose Glaucoma. This will allows for physicians to be able to diagnose Glaucoma in areas where taking large diagnosis equipment is not an option.

GlaucoMed - Title Screen

Figure 4.33 shows the title screen for our case study mobile application **GlaucoMed**. Every mobile application has an initial screen where the user begins when the application starts. This screen is usually referred to as a title screen. In order for our

Figure 4.33: GlaucoMed - Title Screen

framework to fully support the creation of any sort of medical mobile application it had to support at least this operation. In the case of our GlaucoMed this screen was implemented using the finite state machine functionality made available by the framework. On this application every screen is represented as a state in the state machine and is handled by a global finite state machine. In the case of our case study mobile application the title screen allows the user to perform one of two operations: 1) visit the getting started screen, and 3) visit the demo app screen.

GlaucoMed - Getting Started

Figure 4.34 shows the **Getting Started** screen. Most mobile applications which require some training to be used have a getting started section. In this section they walk the user through the functionality of the mobile application in order to quickly train him/her. GlaucoMed is no different from these applications. Our getting started screen walks the user through the different tools available to users inside the mobile

Figure 4.34: GlaucoMed - Getting Started Screen

application. This allows the physician or nurse to quickly get up to speed on the usage of our mobile application.

GlaucoMed - App

Figure 4.35 shows the application screen. GlaucoMed allows physicians to use the camera embedded in their mobile device to capture images of the patient's eye. This is image can then be processed by GlaucoMed and analyzed to detect any symptom of Glaucoma. GlaucoMed automates this task by sending the image through our proposed image extraction component. It is also important to point out that this stage each marker on the mobile device is touch sensitive allowing the physician to query the expert system for additional information or suggestion about common procedures to perform depending on the patient's symptom.

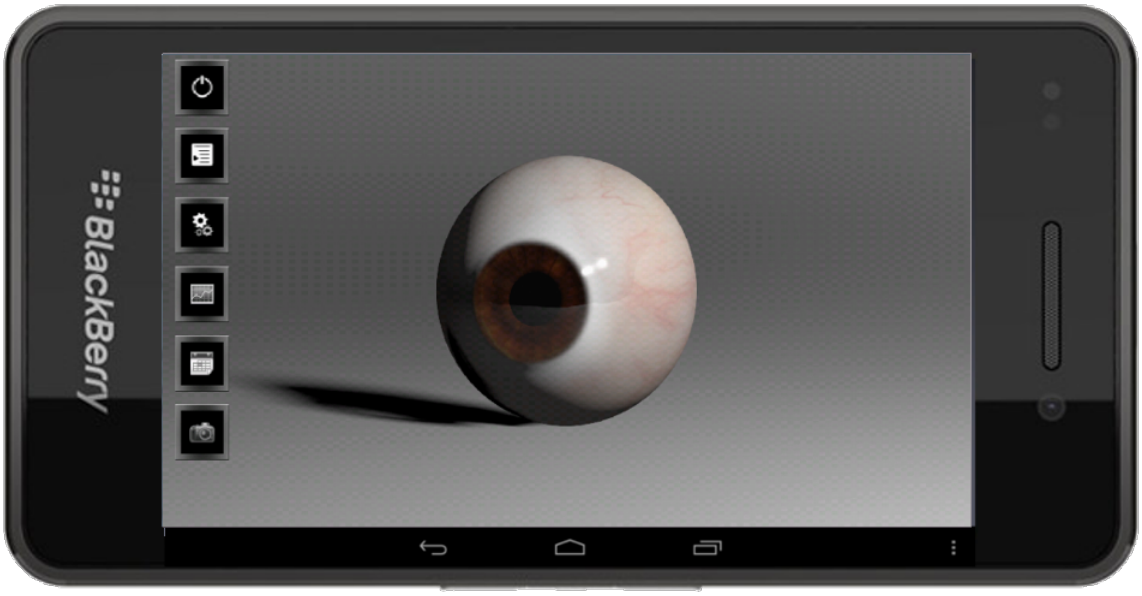


Figure 4.35: GlaucoMed - App Screen

CHAPTER 5

EXPERIMENT RESULTS

5.1 UIML

In this section, we describe the experiments we performed to evaluate our language **UIML**. We first present the design of our experiments, then the results, followed by a discussion of our findings. Since our intent is to reduce the amount effort required for developers to create user interfaces, we use two metrics in our experiments: time and lines of code (**LOC**) required to develop each user interface with and without **UIML**. For our experiments we used a subset of our language we call BUIML, for BlackBery User Interface Modeling Language. This version translates the models generate using out language UIML into development artifacts specific the Blackberry platform.

5.1.1 Procedure and Scenarios

In order to evaluate **UIML**, we developed each of the following user interfaces specifications first without using our approach, and then a second time using our user interface modeling language. We then timed both entire processes from start to completion. The second metric we used was the lines of code (**LOC**) of each version of the user interface, DSL vs. Non-DSL. Since BugIT (App Specification 2) is an attempt to model a user interface being developed by the students of the Game Developers Guild, we decided to use their existing project as a comparison instead of writing

this user interface from scratch ourselves. We believe this gives us a more accurate set of data by comparing our approach with a real ongoing application-development process.

App Specification 1 - Hello World UI: The Hello World UI is a simple user interface which contains all the basic components found in a user interface. The goal is to create a simple and functional interface which supports all the basic functionality of any give user interface. The goal of this application is to showcase all the controls currently available in our **UIML**

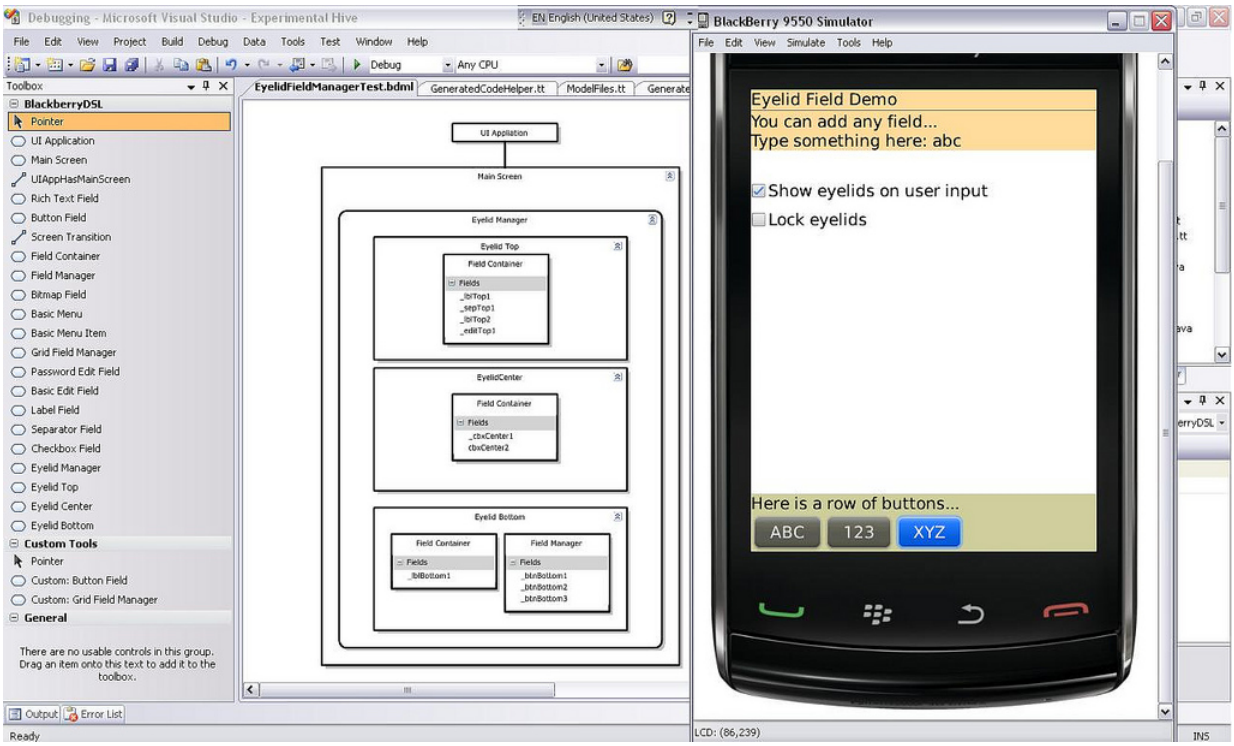


Figure 5.1: Hello World UI App

App Specification 2 - BugIt: BugIt is an application currently under development by members of the Game Developers Guild. The purpose of the application is to teach debugging skill in a manner that is fun an interactive. The application BugIt is composed of four screens: Title Screen, About Screen, Selection Screen, and Bug Screen. The goal of the user in BugIt is to select a problem from a list of bugs

and spot the line with the error. This application contains multiple screen, buttons, table views and images which make it a good test case for exploring the effectiveness of **UIML**.

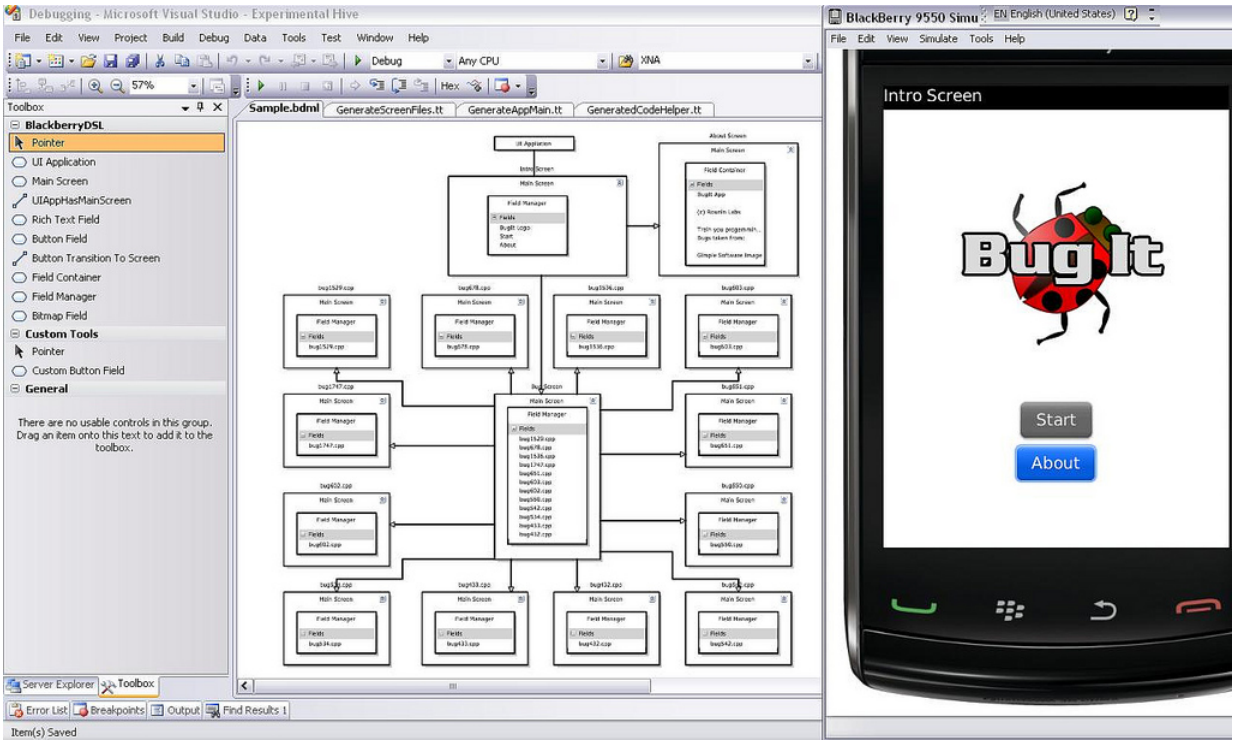


Figure 5.2: BugIt App

5.1.2 Experiment Setup

In this section we present the set up for both or our experiments. The results in lines of code represents the number of lines of code in Java required to complete both the application by hand and the app using **B-UIML**.

Set Up Experiment 1 - Hello World UI: First the app of Hello World UI was implemented by hand, then it was modeled and implemented, both by the author.

Set Up Experiment 2 - BugIt: For the experiment of BugIt, we used an existing project from the Game Developers Guild. We gathered the information of lines of code and time taken from this existing project then we preceded to model the current state of this app and program it using our language. In the case of BugIt the app without our proposed language was developed by the members of the Game Developers Guild while the app implemented using **B-UIML** was written by the author.

5.1.3 Results

In order to perform the experiment, we developed a graphical editor for **U**IML using the Microsoft DSL Tools [26]. In order to show the efforts saved using our approach compared to implementing directly, we have summarized our results in Table 5.1 and Table 5.2. We also implemented a code generator to transform **UIML** models into BlackBerry/Java code. The numbers given represent the two metrics we used when evaluating our language: time and lines of code (**LOC**).

Table 5.1: Effort for creation of Hello World UI using UIML vs. manually developed

App: Hello World UI	**B-UIML**	**Manually Developed**
Dev. Time (Hours)	0.32	0.54
Effort (LOC)	0	753

For each app explored, we show the actual effort of directly implementing the app, and the actual effort of implementing the app using our UI modeling language. In Hello World UI, we could reduce 753 (753 - 0) lines of code (**LOC**) written by the user and, as a result, 100% (753/753) of the amount of work required by the user

Table 5.2: Effort for creation of BugIt using B-UIML vs. manually developed

App: BugIt	**B-UIML**	**Manually Developed**
Dev. Time (Hours)	1.03	10.4
Effort (LOC)	212	4231

was reduced. Table 5.1 also shows a savings of 0.24 (0.56 - 0.32) hours, a 44.4% (0.24/0.54) savings on the time required to develop the app by using **B-UIML** as opposed to implementing it without it. We also present the results of applying our approach to BugIt, a more complex app than Hello World UI. Using our approach as seen in Table 5.2, we were able to reduce 4019 (4231 - 212) lines of code written by the user, and save 94.9% (4019/4231) of the amount lines of code as opposed to implementing the app directly. There was also a significant reduction of 9.37 (10.4 - 1.03) hours required, a savings of 90.09% (9.37/10.4) of the time spent creating the app by using our approach as opposed to implementing the app without it.

5.1.4 Discussion

From Table 5.1 and Table 5.2, we can see that the amount of savings varies greatly depending on the complexity of the app. We attribute the difference in the effort percentage between Hello World UI 100% and BugIt 90.04% to the expressiveness of our language, and the level of complexity of the apps directly. The game of Hello World UI app consists mostly of simple interactions.As a result of this, we were able to model a large set of the user input app representation. In comparison, with a more complex app like BugIt, which requires a lot of implementation of user and application logic, we obtained a better savings of 90.04% fewer lines of code. With

the logic section of **B-UIML**, we were capable of modeling a large section transition between screens in response to user imput.

5.2 Eberos GML

In this section, we describe the experiments we performed to evaluate our language **Eberos GML2D**. We first present the design of our two experiments, then the results, followed by a discussion of our findings. Since our intent is to reduce the amount of both time and work required to develop 2D games, we use two metrics in our experiments: time and lines of code (**LOC**) required to develop each game with and without **Eberos GML2D**. We used Microsoft's XNA game engine for developing the games from scratch. The generated models were also translated into XNA Code as well.

5.2.1 Procedure and Scenarios

In order to evaluate **Eberos GML2D**, we developed each of the following game specifications first without using our approach, and then a second time using our DSL. We then timed both entire processes from start to completion. The second metric we used was the lines of code (**LOC**) of each version of the game, DSL vs. Non-DSL. Since SpaceKatz (Game Specification 2) is an attempt to model a game being developed by the students of SIG-Games at Florida International University (**FIU**), we decided to use their existing project as a comparison instead of writing this game from scratch ourselves. We believe this gives us a more accurate set of data by comparing our approach with a real ongoing game-development process.

Game Specification 1 - Pong: Pong is as simple a 2D game as they come; it consists of two paddles (one for each player located at each end of the screen) and a ball. The ball bounces off the walls and the player's paddle. The goal of the game is to get the ball past the opponent's paddle in order to score.

Game Specification 2 - SpaceKatz: SpaceKatz is a 'Shoot 'em Up' style of game, where the player must pilot through asteroid fields while destroying enemies in the process. This game consists of one main screen with the game logo. Following the logo screen, the player is presented with a menu where he/she can select what options to set or directly start the game. After selecting to start the game, the player is presented with a menu to choose the level or area to play. Once an area is selected the game begins, and the player is presented with the spaceship to pilot. A player can move in all four directions by pressing the keyboard keys (Left, Right, Up, Down); he/she can also shoot missiles by pressing the "Space Bar" key on the keyboard. The game ends when the player loses all of his/her lives.

5.2.2 Experiment Setup

In this section we present the set up for both or our experiments. The results in lines of code represents the number of lines of code in XNA/C# required to complete both the game by hand and the game using **Eberos GML2D**.

Set Up Experiment 1 - Pong: First the game of Pong was implemented by hand, then it was modeled and implemented, both by the main author of this paper.

Set Up Experiment 2 - SpaceKatz: For the experiment of SpaceKatz, we used an existing project from SIG-Games. We gathered the information of lines of code and time taken from this existing project then we preceded to model the current state of this game and program it using our language. In the case of SpaceKatz the game without our proposed language was developed by the members of SIG-Games while the game implemented using **Eberos GML2D** was written by the main author of this paper.

Note: We feel it is important to note that SIG-Games is a group dedicated to training students to become game developers, and that past members have worked on AAA titles such as **Bioshock 2**.

5.2.3 Results

In order to perform the experiment, we developed a graphical editor for **Eberos GML2D** using the Microsoft DSL Tools [26]. In order to show the efforts saved using our approach compared to implementing directly, we have summarized our results in Table 5.3 and Table 5.4. We also implemented a code generator to transform **Eberos GML2D** models into Microsoft XNA code. The numbers given represent the two metrics we used when evaluating our language: time and lines of code (**LOC**).

For each game explored, we show the actual effort of directly implementing the game, and the actual effort of implementing the game using our language. In Pong, we could reduce 39 (131 - 92) lines of code (**LOC**) written by the user and, as a result, 29% (39/131) of the amount of work required by the user was reduced. Table 5.3 also shows a savings of 3 (34 - 31) minutes, a 8.82% (3/34) savings on the time required to

Table 5.3: Effort for creation of Pong using Eberos GML2D vs. manually developed

Game: Pong	**Eberos GML2D**	**Manually Developed**
Dev. Time (Hours)	0.13 Modeling + 0.38 Implementing = 0.51 total	0.56 Implementing
Effort (LOC)	92 User Gen. + 74 XNA Gen. + 1870 Auto Gen. = 2036 total	131 User Generated + 74 XNA Generated = 205 total

develop the game by using **Eberos GML2D** as opposed to implementing it without it. We also present the results of applying our approach to SpaceKatz, a more complex game than Pong. Using our approach as seen in Table 5.4, we were able to reduce 3303 (3822 - 519) lines of code written by the user, and save 86.4% (3303/3822) of the amount lines of code as opposed to implementing the game directly. There was also a significant reduction of 24.67 (30 - 5.33) hours required, a savings of 82.3% (24.67/30) of the time spent creating the game by using our approach as opposed to implementing the game without it.

5.2.4 Discussion

From Table 5.3 and Table 5.4, we can see that the amount of savings varies greatly depending on the complexity of the game. We attribute the difference in the effort percentage between Pong 29% and SpaceKatz 86.4% to the expressiveness of our language, and the level of complexity of the games directly. The game of Pong consists

Table 5.4: Effort for creation of SpaceKatz using Eberos GML2D vs. manually developed

Game: SpaceKatz	**Eberos GML2D**	**Manually Developed**
Dev. Time (Hours)	0.51 Modeling + 4.81 Implementing = 5.32 total	30.0 Implementing
Effort (LOC)	519 User Gen. + 74 XNA Gen. + 5546 Auto Gen. = 6139 total	3822 User Generated + 74 XNA Generated = 3896 total

mostly of interactions. As a result of this, while we were able to model a large set of the user input game representation, there was also a bit of game behavior that could not be modeled with the current version of **Eberos GML2D**. In comparison, with a more complex game like SpaceKatz, which requires a lot of implementation of user and game logic, as well as enemies' AI, we obtained a better savings of 86.4% less lines of code. With the logic section of **Eberos GML2D**, we were capable of modeling a large section of the agent logic for the enemies, menus, and player, something that had to be otherwise implemented without using our approach. Similarly, we were able to model the entities, bullets, asteroids, and pool of entities using **Eberos GML2D**, which had to be implemented for non-DSL version of the game.

5.3 Mobile Modules

In this section, we describe the experiments we performed to evaluate the modules that we developed as part of our development framework. We first present the design of our experiments, then the results, followed by a discussion of our findings. Since our intent is to reduce the amount effort required for developers to create mobile medical applications, we use two metrics in our experiments: time and lines of code (**LOC**) required to develop each mobile medical application with and without our modules.

5.3.1 Procedure and Scenarios

In order to evaluate the modules created, we first developed each of the following user mobile medical applications without using our approach, and then a second time using our medical modules. We then timed both entire processes from start to completion. The second metric we used was the lines of code (**LOC**) of each version of the mobile medical app, Module vs. Non-Module.

App Specification 1 - HairX: HairX is an application that we at FIU have been developing in cooperation with Dr.Neusbaum to aid in the recognition of hair follicles and improve detection of transplant zones. Aside from the image analysis and suggestion component, this mobile app also contains other features such as:

- Patient Management System
- Patient Visit Records

- Billing
- Cloud Support
- Encryption/Decryption of Records.

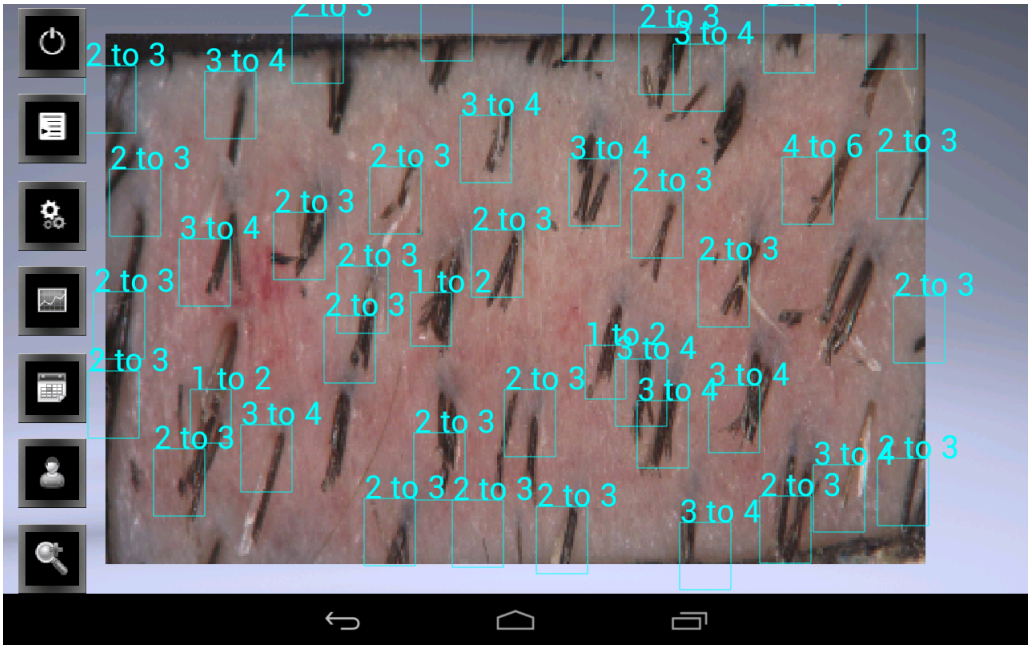


Figure 5.3: HairX App

App Specification 2 - GlaucoMed: GlaucoMed is a research mobile medical application with the purpose of providing users with the ability to diagnose the threat of Gloaucoma simply using a smartphone or any other mobile devices. The goal to provide physicians in areas of low resource with an affordable means to diagnose Glaucoma. As additional support for these physicians, GlaucoMed also provides the following features:

- Patient Management System
- Patient Visit Records
- Billing

- Encryption/Decryption of Records.

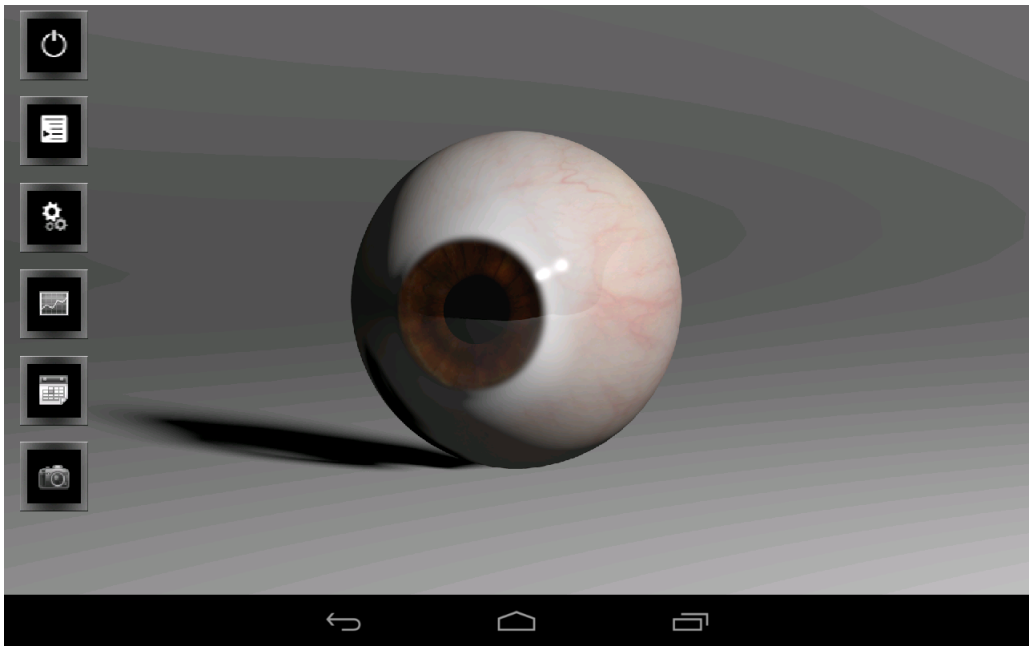


Figure 5.4: GlaucoMed App

5.3.2 Experiment Setup

In this section we present the set up for both or our experiments. The results in lines of code represents the number of lines of code in Java required to complete both the application from scratch and the application using the modules created during this work.

Set Up Experiment 1 - HairX: First the app of Hello World UI was implemented as an average developer would. This means that all features were implemented from the ground up without the use of any rapid development frameworks. Once the app was completed this way the total lines of code and time spent were recorded. Once we had this data we proceeded to create the application once more but this time using all the modules we have created in this work.

Set Up Experiment 2 - GlaucoMed: The same approached that was followed for the experiments performed in HairX were performed with GlaucoMed. This mobile medical applications we first implemented from the ground up without the use of any rapid development frameworks. Once the app was completed this way the total lines of code and time spent were recorded. Once we had this data we proceeded to create the application once more but this time using all the modules we have created in this work.

5.3.3 Results

In order to show the efforts saved using our approach compared to implementing directly, we have summarized our results in Table 5.5 and Table 5.6. The numbers given represent the two metrics we used when evaluating our language: time and lines of code (**LOC**).

Table 5.5: Effort for creation of HairX using Modules vs. manually developed

App: HairX	**Modules**	**Manually Developed**
Dev. Time (Hours)	10.3	162.3
Effort (LOC)	657	10035

Table 5.6: Effort for creation of GlaucoMed using Modules vs. manually developed

App: GlaucoMed	**Modules**	**Manually Developed**
Dev. Time (Hours)	8.4	130.2
Effort (LOC)	562	8332

For each app explored, we show the actual effort of directly implementing the app, and the actual effort of implementing the app using our mobules. In HairX, we could reduce 9378 (10035 - 657) lines of code (**LOC**) written by the user and, as a result, 93.4% (9378/10035) of the amount of work required by the user was reduced. Table 5.5 also shows a savings of 150 (162.3 - 10.3) hours, a 91.8% (150/163.3) savings on the time required to develop the app by using our modules as opposed to implementing the app without them. We also present the results of applying our approach to GlaucoMed. Using our approach as seen in Table **??**, we were able to reduce 7770 (8332 - 562) lines of code written by the user, and save 93.2% (7770/8332) of the amount lines of code as opposed to implementing the app directly. There was also a significant reduction of 121.8 (130.2 - 8.4) hours required, a savings of 93.5% (121.8/130.2) of the time spent creating the app by using our approach as opposed to implementing the app without it.

5.3.4 Discussion

From Table 5.5 and Table 5.6, we can see that there is a significant saving in effort between using the modules developed as past of our work and creating the application from scratch. This is largely due to the fact that part of our approach has been to abstract the concepts that are key to mobile medical applications. It is important to point out that these benefits can only be achieved when developing mobile applications in the medical domain and do not extend to other domains in the mobile space.

CHAPTER 6

CONCLUSION

This chapter provides a summary of the research presented in this dissertation, and discusses future directions in the areas of rapid development and suggestion systems. In the summary, we present the overview of the conducted work, the results of our experiments and the discussion of the extent to which our proposed solutions satisfied our research goals. The future work section describes several topics concentrating on the further exploration of the medical domain and the extension of our rapid development framework for mobile medical applications. This final chapter contains both research summary and possible future work.

6.1 Research Summary

The main objective of the dissertation was to propose a framework to assisting developers of mobile medical application to use during the design and implementation of their application and reduce the overall effort required to complete these applications. The identified research problem was divided into the following sub-problems: 1. Devise a a graphical modeling language that facilitates the specification and development of user interfaces for mobile applications; 2. Devise a graphical modeling language that facilitates the development of mobile graphical applications in mobile devices; and 3. Design and develop a framework for the rapid development of mobile medical applications which provides developers with reliable and high fidelity re-usable modules.

The solution to the first sub-problem lead to the creation of our User Interface Modeling Language (**UIML**). This graphical language abstracts the UI components of mobile devices into graphical constructs allowing developers to create user interfaces for their mobile applications using a simple drag and drop approach. Due to the inherent nature of languages, models generated using UIML can easily be validated for errors. This allows developers to spot problems in their design before it reaches the code. Models generated using **UIML** can be easily translated into ready to use source code which can be directly added to any existing project. This in turn yields a reduction in the time taken to develop the user interface in a mobile medical application. In this dissertation we developed **B-UIML** an implementation of **UIML** aimed at BlackBerry devices.

The solution to the second sub-problem came in the form of **Eberos GML2D**. In order to fully support a wide range of mobile medical application there was the need to devise a way to reduce the effort of creating application which required more interaction and access to the graphic level. Example application include medical simulation apps in which physicians get to interact with a 3D or 2D representation of a human subject. In reduce the effort of creating this game-like applications we developed the Eberos Game Modeling Language. This graphical language allows developers to model their simulation applications using graphical constructs using a simple drag and drop approach. Much like with our work on **UIML** models developed using **Eberos GML** can be validate in order to spot any errors. For this dissertation we created a subset of **Eberos GML** called Eberos Game Modeling Language 2D **Eberos GML2D** aimed at supporting the domain of 2D medical applications.

Finally, our solution to third sub-problem was the design and development of a module-based framework for the development of mobile medical applications. This

framework abstracts concepts from the medical domain and encapsulates them into re-usable modules which developers can then integrate into the development of their apps. For the purpose of these dissertation we have encapsulated the most common components in mobile medical applications. The modules developed were: patient records; patient visit record, billing, image analysis, cloud manager, and procedure suggestion.

We have thus incorporated all these three solution into one single framework we call MOBIle MEDical Rapid Application Development framework **MOBIMED**. By combining all these three solutions we have managed to reduce the effort required to develop the user interface, simulation and other mobile medical applications.

6.2 Future Work

The research presented in this dissertation provides the foundation for developing a Rapid Development Framework for mobile medical applications. Presently, we have tackled a large part of the problem through our contributions presented here but there remains a lot more work to be done. In the area of our UIML there is still a lot of room for improvements. Our implementation B-UIML translates the models into BlackBerry specific code. There still remains the need to develop translators to support other mobile platforms like the Windows Phone, Android, and iPhone. Similarly, much work remains to be done for our Eberos GML. The current version presented in this dissertation only addresses the area of 2D development. While a lot of the components that apply to the development of 3D applications are already in the language, it still lacks many concepts unique to the 3D domain.

Finally, we explored a lot of concepts that are unique and common most common mobile medical application, the current implementation only targets the Android platform. In the future we plan to explore abstracting these components to a higher intermediate level language. This will allows for applications to be expressed in this platform independent language language and then be easily translated into platform specific code.

CHAPTER 7

PUBLICATIONS LIST

7.1 Book Chapters

- Peter J. Clarke, Yali Wu, Andrew A. Allen, Frank Hernandez, Mark Allison, and Robert France. Towards dynamic semantics for synthesizing domain-specific models. In *Formal and Practical Aspects of Domain-Specific Languages: Recent Developments*. IGI Global, 2012

- Frank E. Hernandez and Francisco R. Ortega. Reducing video game creation effort with eberos gml2d. In *In Game Development Tools*. K Peters/CRC, 2011

7.2 Journals

- Frank Hernandez and S.S. Iyengar. Mobimed: A modeling environment for building software tools for medical mobile applications. *International Journal of Multimedia Data Engineering and Management (IJMDEM) (Submitted)*, 2013

- Yali Wu, Frank Hernandez, Peter J. Clarke, and Robert B. France. A model driven approach to realizing user-centric communication services. *SPE*, 2011

- Christine Lisetti, Emmanuel Pozzo, Marie Lucas, Frank Hernandez, and W. Kurtines W. Silverman. How to program in the second life virtual world: A case study for anxiety disorders. *Journal of Cyberpsychology and Behavior*, 2009

- S. S. Iyengar, Wei Zeng, Frank Hernandez, Bernard P. Nusbaum, and Paul Rose. Context based algorithmic framework for identifying and classifying embedded images of follicle units. In *Computerized Medical Imaging and Graphics (Submitted)*, 2013

- Peter J. Clarke, Jairo Pava, Debra Davis, Frank Hernandez, and Tariq M. King. A coupling-guided cluster analysis approach to reengineer the modularity of object-oriented systems. In *Using WReSTT in SE courses: an empirical study*, pages 30–7–312. SIGCSE, 2012

- Yali Wu, Frank Hernandez, Peter J. Clarke, and Robert B. France. A dsml for coordinating user-centric communication services. In *Using WReSTT in SE courses: an empirical study*. COMPSAC11 (2011 35th Annual IEEE International Computer Software and Applications Conference), 2011

- Christine Lisetti, Emmanuel Pozzo, Marie Lucas, Frank Hernandez. W. Silverman, and W. Kurtines. Programming in the second life virtual world. In *In Proceedings of the 14th Annual Conference on CyberTherapy and CyberPsychology Conference - Studies in Health Technology and Informatics (SHTI)*. IO Press: Amsterdam, 2009

- Yali Wu, Andrew A. Allen, Frank Hernandez, Yingbo Wang, and Peter J. Clarke. A user-centric communication middleware for cvm. In *IASTED International Conference on Software Engineering and Applications*, pages 210–215. SEA, 2008

7.4 Workshops

- Yali Wu, Frank Hernandez, Francisco Ortega, Peter J. Clarke, and Robert France. Measuring the effort for creating and using domain-specific models. In *Proceedings of the 10th SPLASH Workshop on Domain-Speci?c Modeling*. DSM, 2010

- Frank Hernandez and Peter J. Clarke. Towards integration of policies into dsmls. In *Proceedings of the 10th SPLASH Workshop on Domain-Specific Modeling*. DSM, 2011

- Frank E. Hernandez and Francisco R. Ortega. Eberos gml2d: A graphical domain-specific language for modeling 2d video games. In *Proceedings of the 10th SPLASH Workshop on Domain-Speci?c Modeling*. DSM, 2010

7.5 Posters

- Frank Hernandez, Yali Wu, Robert France, and Peter J. Clarke. Wf-cml: Dsml for coordinating user-centric communication services. In *International Workshop On Formalization Of Modeling Languages*. DSM, 2010

- Francisco Ortega, Frank Hernandez, Armando Barreto, Naphtali Rishe, and Malek Adjouadi. Exploring modeling language for multi-touch systems using pretirnet. In *ACM International Conference on Interactive Tabletops And Surfaces*. ITS, 2013

CHAPTER 8

AWARDS

- **Miami's Got Tech Award:** Award given by the Greater Miami Chamber of Commerce for the best mobile application during the Miami's Got Tech 2012.
- **BlackBerry Crystal Award:** Award provided by BlackBerry to outstanding developers. 2012
- **Best Android App:** Obtained at the 24 hour hackathon sponsored by AT&T in December 2012 for developing the best Android application from among all the participants. This event included around 100 participants from the South Florida area.
- **Best in Show Sphero SDK:** Obtained at the 24 hour hackathon sponsored by AT&T in December 2012 for building best application using the Sphero SDK from among all the participants. This event included around 100 participants from the South Florida area.
- **Best Overall App (2n Place):** Obtained at the 24 hour hackathon sponsored by AT&T in December 2012 for building the second best application from among all the participants. This event included around 100 participants from the South Florida area.
- **Best Use of Sphero SDK:** Obtained at the 24 hour hackathon sponsored by AT&T in August 2012 for building the best application using the Sphero SDK from among all the participants. This event included around 150 participants from the South Florida area.

- **Graduate Assistance in Areas of National Need (GAANN) Fellowship.** 2008-2013

CHAPTER 9

ACHIEVEMENTS

- **Game Developers Guild:** Founded the first incubator for game developers in the entire South Florida area (http://www.GameDevelopersGuild.com). The Game Developers Guild aims at providing students as well as local developers looking to enter the game development market, with the training and mentoring necessary to succeed. Since its foundation the Game Developers has had the following successes:
  - **60+ Games Developed:** Since its creation members of the Game Developers Guild have created and taken to market over 60 games. Their games have been published in Google Play, BlackBerry World, Amazon and Windows Phone Store.
  - **Workshops and Presentations:** The guild has hosted 6 workshops and developer networking events to enhance the level of tech and game development in the South Florida Area.
  - **Incubated Companies**: The Oneironaght Games Game Studio was successfully incubated at the Game Developers Guild. Since its launch in July 2012 OG has managed to successfully take to market their game "Find the Gnomes" for the Windows store.
- **Testimonials of Members of the Game Developers Guild:**
  - **GDG Member A:** "Mobile optimization, cross platform development, rapid prototyping, competitive development, project scoping, project budgeting, 3D modeling , 2D texturing, marketing, customer support and

much more. These are the skills that our students are never exposed to and I would have never been able to get as part of a standard college education from my degree of Computer Science and it shows in my professional interactions and knowledge set by allowing me to do more and have experience interacting with people from multiple specializations. To the Guild Members, something difficult just means we have not done that yet."

- **GDG Member B:** "There is something interesting that happens to you inside the Guild. You won't realize it. You won't even see it coming. You start learning effortlessly. You might be working on a group project or on your own personal game. Then you run into a problem. It is a self discovered problem, so you have this passion to solve it. Thus you ask the other people who are working on their own projects. They might know the answer and give you a very intense introduction into the solution that forces you to reverse engineer it and learn it. Or they won't know, but then, you now have another person interested in the problem; so the two of you eagerly search for the solution until you figure out a way to solve it. That is something you won't get anywhere else, not school. In schools you usually get an assignment that is hardly ever related to the lectures and you have no real incentive to solve it, besides your grade. Usually these assignments are trivial and not necessarily realistic approaches to programming problems, they just exist to brute-force-introduce you into concepts. That essence of discovery and learning that is missing in our schools. That is what you get at the Guild."
- **GDG Member C:** "The Game Developers Guild has given me the resources to succeed as a computer science student. Not only has it forced me to push my programming skills to the limit, it has also exposed me to

areas that aren't generally (focused on) in CS programs. It has helped to make my education well rounded by exposing me to artistic ideologies, the psychology of game design, and the business aspects of game development. Programming for games has reinforced many of the skills that I learned in school and introduced me to languages and programming ideas that would have been overlooked in a general C.S. education, such as graphics programming (CG, HLSL), parallel programming, physics and 3d programming. The Game Developers Guild's drive to build a thriving local indie development scene has brought me in contact with many great people and showed me many of the valuable opportunities the future holds."

- **Global Game Jam 2012:** Successfully managed to bring the Global Game Jam to Miami, Florida for the first time in the history of the event in January 2012. The Global Game Jam is a 48 hours international game development event that takes place all over the world on the last weekend of January. For this event I secured sponsorship from Blizzard, one of the largest video game development companies in the world.

- **Game Jam 4 Life:** Created a Game Jam 4 Life a 24 hour game jam where developers gather and program games while raising funds for the Miami Children's Hospital.

- **Global Game Jam 2013:** Managed to successfully bring back to Miami the Global Game Jam for its second year in a row. This time we had a record-breaking number of participants as well as games developed.

- **Game Day:** Created Game Day a networking event for local game developers with the goal of increasing communication between local developers and grow the South Florida technology ecosystem. This event has drawn in large game

developer studios like Nuclei3D which has offered internships to a few FIU students.

BIBLIOGRAPHY

[1] Sos S. Agaian, Karen P. Lentz, and Artyom M. Grigoryan. A new measure of image enhancement. In *IASTED International Conference on Signal Processing and Communication*, 2000.

[2] Sos S. Agaian, Karen A. Panetta, and Artyom M. Grigoryan. Transform-based image enhancement algorithms with performance measure. *IEEE Transactions on Image Processing*, 10(3):367–382, 2001.

[3] S.S. Agaian, K. Panetta, and A.M. Grigoryan. Transform-based image enhancement algorithms with performance measure. *Image Processing, IEEE Transactions on*, 10(3):367–382, 2001.

[4] Sabzali Aghagolzadeh and Okan K. Ersoy. Transform image enhancement. *Optical Engineering*, 31(3):614–626, 1992.

[5] S. Ando. Image field categorization and edge/corner detection from gradient covariance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(2):179–190, 2000.

[6] Kyungim Baek, Bruce A. Draper, J. Ross Beveridge, and Kai She. Pca vs. ica: a comparison on the feret data set. In *in Proc. of the 4th International Conference on Computer Vision, ICCV02*, pages 824–827, 2002.

[7] M.S. Bartlett, Javier R. Movellan, and T.J. Sejnowski. Face recognition by independent component analysis. *Neural Networks, IEEE Transactions on*, 13(6):1450–1464, Nov.

[8] Fredrik Bergholm. Edge focusing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(6):726–741, 1987.

[9] J.C. Bezdek, R.A. Chandrasekhar, and Y. Attikouzel. A geometric approach to edge detection. *Fuzzy Systems, IEEE Transactions on*, 6(1):52–75, 1998.

[10] Enzo Bolis, Christian Zerbi, Pierre Collet, Jean Louchet, and Evelyne Lutton. A gp artificial ant for image processing: preliminary experiments with easea. In Julian Miller, Marco Tomassini, PierLuca Lanzi, Conor Ryan, AndreaG.B. Tettamanzi, and WilliamB. Langdon, editors, *Genetic Programming*, volume 2038 of *Lecture Notes in Computer Science*, pages 246–255. Springer Berlin Heidelberg, 2001.

[11] K. Bowyer, C. Kranenburg, and S. Dougherty. Edge detector evaluation using empirical roc curves. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 1, pages –359 Vol. 1, 1999.

[12] M. J. Brooks. Rationalizing edge detectors. *j-CGIP*, 8(2):277–285, oct 1978.

[13] Bruce G. Buchanan and Edward H. Shortliffe. *Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project (The Addison-Wesley series in artificial intelligence)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984.

[14] Mat Buckland. *Programming Game AI by Example*. Wordware Pub, Plano, Tx, 1st edition, 2005.

[15] P.J. Burt and E.H. Adelson. The laplacian pyramid as a compact image code. *Communications, IEEE Transactions on*, 31(4):532–540, 1983.

[16] Busby, Jason, Zak Parrish, and Jeff Wilson. *Mastering Unreal Technology Volume 1: the Art of Level Design*, volume 1. Sams Publishing, Indianapolis, 2009.

[17] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, June 1986.

[18] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, 1986.

[19] T. Carron and P. Lambert. Color edge detector using jointly hue, saturation and intensity. In *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*, volume 3, pages 977–981 vol.3, 1994.

[20] Heather M. Chandler. *The Game Production Handbook*. Infinity Science, Hingham, Mass, 2009.

[21] G. Chen and Y.H. Hong Yang. Edge detection by regularized cubic b-spline fitting. *Systems, Man and Cybernetics, IEEE Transactions on*, 25(4):636–643, 1995.

[22] Se-Hak Chun and Yoon-Joo Park. A new hybrid data mining technique using a regression case based reasoning: Application to financial forecasting. *Expert Systems with Applications*, 31(2):329 – 336, 2006.

[23] William J. Clancey. From guidon to neomycin and heracles in twenty short lessons: Orn final report 1979-1985. *AI Magazine*, 7(3):40–60, 1986.

[24] Peter J. Clarke, Jairo Pava, Debra Davis, Frank Hernandez, and Tariq M. King. A coupling-guided cluster analysis approach to reengineer the modularity of object-oriented systems. In *Using WReSTT in SE courses: an empirical study*, pages 30–7–312. SIGCSE, 2012.

[25] Peter J. Clarke, Yali Wu, Andrew A. Allen, Frank Hernandez, Mark Allison, and Robert France. Towards dynamic semantics for synthesizing domain-specific models. In *Formal and Practical Aspects of Domain-Specific Languages: Recent Developments*. IGI Global, 2012.

[26] Steve Cook, Gareth Jones, Stuart Kent, and Aran Cameron Wills. *Domain-specific Development with Visual Studio DSL Tools*. Addison-Wesley, Upper Saddle River, NJ, 2007.

[27] Silvano Di Zenzo. A note on the gradient of a multi-image. *Comput. Vision Graph. Image Process.*, 33(1):116–125, January 1986.

[28] Jeroen Dobbe. A domain-specific language for computer games. Master's thesis, Delft University of Technology, 2007.

[29] R.D. Dony and S. Wesolkowski. Edge detection on color images using rgb vector angles. In *Electrical and Computer Engineering, 1999 IEEE Canadian Conference on*, volume 2, pages 687–692 vol.2, 1999.

[30] Bruce A. Draper, Kyungim Baek, Marian Stewart Bartlett, and J. Ross Beveridge. Recognizing faces with pca and ica. In *COMPUTER VISION AND IMAGE UNDERSTANDING, SPECIAL ISSUE ON FACE RECOGNITION*, pages 115–137, 2003.

[31] Jiang Duan and Guoping Qiu. Novel histogram processing for colour image enhancement. In *Multi-Agent Security and Survivability, 2004 IEEE First Symposium on*, pages 55–58, 2004.

[32] Marc Ebner. On the evolution of edge detectors for robot vision using genetic programming. In Horst-Michael Groß, editor, *Workshop SOAVE '97 - Selbstorganisation von Adaptivem Verhalten, VDI Reihe 8 Nr. 663*, pages 127–134, Düsseldorf, 1997. VDI Verlag.

[33] Mehdi Fesharaki, Hossein Shirazi, Akram Bakhshi, and AWT_TAG. knowledge acquistion from database of information management and documentation softwares by datamining techniques. *Journal of Information Processing and Management*, 26, 2011.

[34] J. Fortuna, P. Quick, and D. Capson. A comparison of subspace methods for accurate position measurement. In *Image Analysis and Interpretation, 2004. 6th IEEE Southwest Symposium on*, pages 16–20, March.

[35] J. Fortuna, D. Schuurman, and D. Capson. A comparison of pca and ica for object recognition under varying illumination. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 3, pages 11–15 vol.3.

[36] Werner Frei and Chung-Ching Chen. Fast boundary detection: A generalization and a new algorithm. *Computers, IEEE Transactions on*, C-26(10):988–998, 1977.

[37] Keinosuke Fukunaga. *Introduction to statistical pattern recognition (2nd ed.)*. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[38] Andre W B Furtado, Andre L M Santos, and Geber L Ramalho. A computer games software factory and edutainment platform for microsoft .net. *SB Games 2007*, pages 1–29, Jun 2007.

[39] AWB Furtado and AL de Medeiros Santos. Sharpludus: improving game development experience through software factories and domain-specific languages. *Universidade Federal de Pernambuco (UFPE) Mestrado em Ciência da Computação centro de Informática (CIN)*, 2006.

[40] AWB Furtado and ALM Santos. Using domain-specific modeling towards computer games development industrialization. *6th OOPSLA Workshop on Domain-Specific Modeling (DSM'06)*, page 1, 2006.

[41] AWB Furtado and ALM Santos. Extending visual studio .net as a software factory for computer games development in the .net platform. *2nd International Conference on Innovative Views of .NET Technologies (IVNET06)*, 2007.

[42] Marcin Gajzler. Text and data mining techniques in aspect of knowledge acquisition for decision support system in construction industry. *TECHNOLOGICAL AND ECONOMIC DEVELOPMENT OF ECONOMY*, 16(2), 2010.

[43] S. Ghosal and R. Mehrotra. Detection of composite edges. *Image Processing, IEEE Transactions on*, 3(1):14–25, 1994.

[44] P.H. Gregson. Using angular dispersion of gradient direction for detecting edge ribbons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(7):682–696, 1993.

[45] Jacob Habgood, Mark Overmars, and Phil Wilson. *The Game Maker's Apprentice: Game Development for Beginners*. Apress, Berkeley, CA, 2006.

[46] R.M. Haralick. Digital step edges from zero crossing of second directional derivatives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-6(1):58–68, 1984.

[47] Christopher Harris, , Christopher Harris, Bernard Buxton, and Wce Bt. Evolving edge detectors with genetic programming, 1996.

[48] Harro. Reflectance Based Edge Classification. In *Visual Interfaces*, 1999.

[49] Mark Hedley and Hong Yan. Segmentation of color images using spatial and color space information. *Journal of Electronic Imaging*, 1(4):374–380, 1992.

[50] Frank Hernandez and Peter J. Clarke. Towards integration of policies into dsmls. In *Proceedings of the 10th SPLASH Workshop on Domain-Specific Modeling*. DSM, 2011.

[51] Frank Hernandez and S.S. Iyengar. Mobimed: A modeling environment for building software tools for medical mobile applications. *International Journal of Multimedia Data Engineering and Management (IJMDEM) (Submitted)*, 2013.

[52] Frank Hernandez, Yali Wu, Robert France, and Peter J. Clarke. Wf-cml: Dsml for coordinating user-centric communication services. In *International Workshop On Formalization Of Modeling Languages*. DSM, 2010.

[53] Frank E. Hernandez and Francisco R. Ortega. Eberos gml2d: A graphical domain-specific language for modeling 2d video games. In *Proceedings of the 10th SPLASH Workshop on Domain-Speci?c Modeling*. DSM, 2010.

[54] Frank E. Hernandez and Francisco R. Ortega. Reducing video game creation effort with eberos gml2d. In *In Game Development Tools*. K Peters/CRC, 2011.

[55] Frank Hoonakker, Nicolas Lachiche, Alexandre Varnek, and Alain Wagner. A representation to apply usual data mining techniques to chemical reactions - illustration on the rate constant of s_n2 reactions in water. *International Journal on Artificial Intelligence Tools*, 20(2):253–270, 2011.

[56] L.A. Iverson and S.W. Zucker. Logical/linear operators for image curves. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(10):982–996, 1995.

[57] S. S. Iyengar, Wei Zeng, Frank Hernandez, Bernard P. Nusbaum, and Paul Rose. Context based algorithmic framework for identifying and classifying embedded images of follicle units. In *Computerized Medical Imaging and Graphics (Submitted)*, 2013.

[58] Anil K. Jain. *Fundamentals of digital image processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[59] I. T. Jolliffe. *Principal Component Analysis*. Springer Verlag, New York, 2002.

[60] Allan L. Egbert Jr. and R. C. Lacher. Building emycin expert systems from raw data sources. In *IC-AI*, pages 571–573, 1999.

[61] Hazim Kemal Ekenel and Bülent Sankur. Multiresolution face recognition. *Image Vision Comput.*, 23(5):469–477, May 2005.

[62] III Kingsland, LawrenceC., DonaldA.B. Lindberg, and GordonC. Sharp. Ai/rheum. *Journal of Medical Systems*, 7:221–227, 1983.

[63] M. Kirby and L. Sirovich. Application of the karhunen-loeve procedure for the characterization of human faces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(1):103–108, Jan.

[64] M. Kisworo, S. Venkatesh, and G. West. Modeling edges at subpixel accuracy using the local energy approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(4):405–410, April 1994.

[65] Myung Ko and Kweku-Muata Osei-Bryson. Analyzing the impact of information technology investments using regression and data mining techniques. *Journal of Enterprise Information Management*, 19(4):403–417, 2006.

[66] Robert Kogan, Sos Agaian, and Karen Panetta Lentz. Visualization using rational morphology and zonal magnitude-reduction, 1998.

[67] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems)*. A Bradford Book, 1 edition, December 1992.

[68] L. E. Krueger. Reconciling fechner and stevens - toward a unified psychophysical law. *Behav Brain Sci Behav Brain Sci*, 12(2):251–267, 1989.

[69] C. A. Kulikowski and S. M Weiss. Representation of expert knowledge for consultation: The casnet and expert projects. *Artificial Intelligence in Medicine*, 1982.

[70] A. Laine, J. Fan, and Wuhai Yang. Wavelets for contrast enhancement of digital mammography. *Engineering in Medicine and Biology Magazine, IEEE*, 14(5):536–550, 1995.

[71] Tzong-Huei Lin and Tsair Kao. Adaptive local contrast enhancement method for medical images displayed on a video monitor. *Medical Engineering and Physics*, 22(2):79 – 87, 2000.

[72] Christine Lisetti, Emmanuel Pozzo, Marie Lucas, Frank Hernandez, and W. Kurtines W. Silverman. How to program in the second life virtual world: A case study for anxiety disorders. *Journal of Cyberpsychology and Behavior*, 2009.

[73] Christine Lisetti, Emmanuel Pozzo, Marie Lucas, Frank Hernandez. W. Silverman, and W. Kurtines. Programming in the second life virtual world. In *In Proceedings of the 14th Annual Conference on CyberTherapy and CyberPsychology Conference - Studies in Health Technology and Informatics (SHTI)*. IO Press: Amsterdam, 2009.

[74] Chengjun Liu. Enhanced independent component analysis and its application to content based face image retrieval. *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, 34:1117–1127, 2004.

[75] Chengjun Liu and H. Wechsler. Independent component analysis of gabor features for face recognition. *Neural Networks, IEEE Transactions on*, 14(4):919–928, July.

[76] Chengjun Liu and Harry Wechsler. Comparative assessment of independent component analysis (ica) for face recognition. In *International Conference on Audio and Video Based Biometric Person Authentication*, pages 22–24, 1999.

[77] Xiaoming Liu, J. Tang, and Xiaolong Zhang. A multiscale image enhancement method for calcification detection in screening mammograms. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 677–680, 2009.

[78] J. Luter. Rao, c. r.: Linear statistical inference and its applications, 2. aufl. john wiley and sons, new york-london-sydney-toronto 1973. xx, 625 s., 11.25. *Biometrische Zeitschrift*, 17(8):526–526, 1975.

[79] Raul Machuca and Alton L. Gilbert. Finding edges in noisy scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(1):103–111, 1981.

[80] D. Marr and E. Hildreth. Theory of Edge Detection. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 207(1167):187–217, 1980.

[81] J.-B. Martens. Local orientation analysis in images by means of the hermite transform. *Image Processing, IEEE Transactions on*, 6(8):1103–1116, 1997.

[82] A.M. Martinez and Manli Zhu. Where are linear feature extraction methods applicable? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(12):1934–1944, Dec.

[83] T. Martiriggiano, M. Leo, T. DOrazio, and A. Distante. Face recognition by kernel independent component analysis. In Moonis Ali and Floriana Esposito, editors, *Innovations in Applied Artificial Intelligence*, volume 3533 of *Lecture Notes in Computer Science*, pages 55–58. Springer Berlin Heidelberg, 2005.

[84] P. Meer and B. Georgescu. Edge detection with embedded confidence. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(12):1351–1365, 2001.

[85] Perry L. Miller. Attending: Critiquing a physician's management plan. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(5):449–461, 1983.

[86] J A Mitchell, S L Davenport, M A Hefner, and M M Shei. Use of an expert model to test diagnostic criteria in charge syndrome. *Journal of Medical Systems*, 9(5-6):425–436, 1985.

[87] B. Moghaddam. Principal manifolds and probabilistic subspaces for visual recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(6):780–788, Jun.

[88] Ali Moghaddamzadeh, David Goldman, and Nikolaos G. Bourbakis. A fuzzy-like approach for smoothing and edge detection in color images. *IJPRAI*, 12(6):801–816, 1998.

[89] H. Murase and S.K. Nayar. Visual Learning and Recognition of 3D Objects from Appearance. *International Journal on Computer Vision*, 14(1):5–24, Jan 1995.

[90] Ehsan Nadernejad, Sara Sharifzadeh, and Hamid Hassanpour. Edge detection techniques: Evaluations and comparison. *Applied Mathematical Sciences*, 2(31):1507–1520, 2008.

[91] V.S. Nalwa and T.O. Binford. On detecting edges. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):699–714, 1986.

[92] Alejandra A. Segura Navarrete, Christian Vidal-Castro, Víctor Hugo Menéndez-Domínguez, Pedro G. Campos, and Manuel E. Prieto. Using data mining techniques for exploring learning object repositories. *The Electronic Library*, 29(2):162–180, 2011.

[93] S.K. Nayar, S.A. Nene, and H. Murase. Subspace methods for robot vision. *Robotics and Automation, IEEE Transactions on*, 12(5):750–758, Oct.

[94] A.N. Netravali and B. Prasada. Adaptive quantization of picture signals using spatial masking. *Proceedings of the IEEE*, 65(4):536–548, 1977.

[95] Rainakant Nevatia and K. Ramesh Babu. Linear feature extraction and description. In *Proceedings of the 6th international joint conference on Artificial intelligence - Volume 2*, IJCAI'79, pages 639–641, San Francisco, CA, USA, 1979. Morgan Kaufmann Publishers Inc.

[96] Jeannie Novak. *Game Development Essentials*. Delmar/Cengage Learning, New York, 2010.

[97] K. Ohba and K. Ikeuchi. Detectability, uniqueness, and reliability of eigen windows for stable verification of partially occluded objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(9):1043–1047, Sep.

[98] Francisco Ortega, Frank Hernandez, Armando Barreto, Naphtali Rishe, and Malek Adjouadi. Exploring modeling language for multi-touch systems using pretirnet. In *ACM International Conference on Interactive Tabletops And Surfaces*. ITS, 2013.

[99] R. S. Patil. Causal representation of patient illness for electrolyte and acid-base diagnosis. Technical report, Cambridge, MA, USA, 1981.

[100] H. E. Pople. Caduceus: An experimental expert system for medical diagnosis. *MIT Press*, 5(5):67–80, 1986.

[101] G. Ramponi. Edge extraction by a class of second-order nonlinear filters. *Electronics Letters*, 22(9):482–484, 1986.

[102] Gner S. Robinson. Edge detection by compass gradient masks. *Computer Graphics and Image Processing*, 6(5):492 – 501, 1977.

[103] Andrew Rollings and Ernest Adams. *Andrew Rollings and Ernest Adams on Game Design.* New Riders, London, 2003.

[104] Azriel Rosenfeld and M. Thurston. Edge and curve detection for visual scene analysis. *Computers, IEEE Transactions on*, C-20(5):562–569, 1971.

[105] C.A. Rothwell, J.L. Mundy, W. Hoffman, and V.-D. Nguyen. Driving vision by topology. In *Computer Vision, 1995. Proceedings., International Symposium on*, pages 395–400, 1995.

[106] Stuart J Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall/Pearson Education, Upper Saddle River, NJ, 2003.

[107] H. S. Sahambi and K. Khorasani. A neural-network appearance-based 3-d object recognition using independent component analysis. *Trans. Neur. Netw.*, 14(1):138–149, January 2003.

[108] L. Shafarenko, M. Petrou, and J. Kittler. Automatic watershed segmentation of randomly textured color images. *Image Processing, IEEE Transactions on*, 6(11):1530–1544, 1997.

[109] K. Sam Shanmugam, Fred M. Dickey, and James A. Green. An optimal frequency domain filter for edge detection in digital pictures. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-1(1):37–49, 1979.

[110] Philip Shapira and Jan Youtie. Measures for knowledge-based economic development: Introducing data mining techniques to economic developers in the state of georgia and the us south. *Technological Forecasting and Social Change*, 73(8):950 – 965, 2006.

[111] Akira Shiozaki. Edge extraction using entropy operator. *Computer Vision, Graphics, and Image Processing*, 36(1):1 – 9, 1986.

[112] Stephen M. Smith and J. Michael Brady. Susan: A new approach to low level image processing. *Int. J. Comput. Vision*, 23(1):45–78, May 1997.

[113] D.A. Socolinsky and A. Selinger. A comparative analysis of face recognition performance with visible and thermal infrared imagery. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 4, pages 217–222 vol.4.

[114] Martin Stahl, Til Aach, Thorsten M. Buzug, Sabine Dippel, and Ulrich Neitzel. Noise-resistant weak-structure enhancement for digital radiography. pages 1406–1417, 1999.

[115] William R. Swartout. Xplain: a system for creating and explaining expert consulting programs. *Artificial Intelligence*, 21(3):285 – 325, 1983.

[116] Ali J. Tabatabai and Owen Robert Mitchell. Edge location to subpixel values in digital imagery. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 188–201, 1984.

[117] M. Tabb and N. Ahuja. Multiscale image segmentation by integrated edge and region detection. *Trans. Img. Proc.*, 6(5):642–655, May 1997.

[118] P.E. Trahanias and A.N. Venetsanopoulos. Color edge detection using vector order statistics. *Image Processing, IEEE Transactions on*, 2(2):259–264, 1993.

[119] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *J. Cognitive Neuroscience*, 3(1):71–86, January 1991.

[120] M.A. Vicente, P.O. Hoyer, and A. Hyvarinen. Equivalence of some common linear feature extraction techniques for appearance-based object recognition tasks. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(5):896–900, May.

[121] M.Asuncin Vicente, Cesar Fernndez, Oscar Reinoso, and Luis Pay. 3d object recognition from appearance: Pca versus ica approaches. In Aurlio Campilho and Mohamed Kamel, editors, *Image Analysis and Recognition*, volume 3211 of *Lecture Notes in Computer Science*, pages 547–555. Springer Berlin Heidelberg, 2004.

[122] Jun Wang and Ying Tan. A novel genetic programming based morphological image analysis algorithm. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 979–980, New York, NY, USA, 2010. ACM.

[123] Yali Wu, Andrew A. Allen, Frank Hernandez, Yingbo Wang, and Peter J. Clarke. A user-centric communication middleware for cvm. In *IASTED International Conference on Software Engineering and Applications*, pages 210–215. SEA, 2008.

[124] Yali Wu, Frank Hernandez, Peter J. Clarke, and Robert B. France. A dsml for coordinating user-centric communication services. In *Using WReSTT in SE courses: an empirical study*. COMPSAC11 (2011 35th Annual IEEE International Computer Software and Applications Conference), 2011.

[125] Yali Wu, Frank Hernandez, Peter J. Clarke, and Robert B. France. A model driven approach to realizing user-centric communication services. *SPE*, 2011.

[126] Yali Wu, Frank Hernandez, Francisco Ortega, Peter J. Clarke, and Robert France. Measuring the effort for creating and using domain-specific models. In *Proceedings of the 10th SPLASH Workshop on Domain-Speci?c Modeling*. DSM, 2010.

[127] Jian Yang, D. Zhang, and Jing-Yu Yang. Is ica significantly better than pca for face recognition? In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 198–203 Vol. 1, Oct.

[128] Yi Yang. Colour edge detection and segmentation using vector analysis. Master's Thesis, University of Toronto, 1995.

[129] PongC. Yuen and J.H. Lai. Independent component analysis of face images. In Seong-Whan Lee, HeinrichH. Blthoff, and Tomaso Poggio, editors, *Biologically Motivated Computer Vision*, volume 1811 of *Lecture Notes in Computer Science*, pages 545–553. Springer Berlin Heidelberg, 2000.

[130] Yang Zhang and Peter I. Rockett. Evolving optimal feature extraction using multi-objective genetic programming: a methodology and preliminary study on edge detection. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 795–802, New York, NY, USA, 2005. ACM.

[131] Oscar A. Zuniga and R.M. Haralick. Integrated directional derivative gradient operator. *Systems, Man and Cybernetics, IEEE Transactions on*, 17(3):508–517, 1987.

VITA

FRANK HERNANDEZ

2013	Doctoral Candidate in Computer Science Florida International University Miami, Florida
2008	Master of Science in Computer Science Florida International University Miami, Florida
2006	Bachelor of Science in Computer Science Florida International University Miami, Florida

PUBLICATIONS AND PRESENTATIONS

Frank Hernandez and S.S. Iyengar. Mobimed: A modeling environment for building software tools for medical mobile applications. *International Journal of Multimedia Data Engineering and Management (IJMDEM) (Submitted)*, 2013

S. S. Iyengar, Wei Zeng, Frank Hernandez, Bernard P. Nusbaum, and Paul Rose. Context based algorithmic framework for identifying and classifying embedded images of follicle units. In *Computerized Medical Imaging and Graphics (Submitted)*, 2013

Frank E. Hernandez and Francisco R. Ortega. Reducing video game creation effort with eberos gml2d. In *In Game Development Tools*. K Peters/CRC, 2011

Francisco Ortega, Frank Hernandez, Armando Barreto, Naphtali Rishe, and Malek Adjouadi. Exploring modeling language for multi-touch systems using pretirnet. In *ACM International Conference on Interactive Tabletops And Surfaces*. ITS, 2013

Peter J. Clarke, Yali Wu, Andrew A. Allen, Frank Hernandez, Mark Allison, and Robert France. Towards dynamic semantics for synthesizing domain-specific models.

In *Formal and Practical Aspects of Domain-Specific Languages: Recent Developments*. IGI Global, 2012

Yali Wu, Frank Hernandez, Peter J. Clarke, and Robert B. France. A model driven approach to realizing user-centric communication services. *SPE*, 2011

Christine Lisetti, Emmanuel Pozzo, Marie Lucas, Frank Hernandez, and W. Kurtines W. Silverman. How to program in the second life virtual world: A case study for anxiety disorders. *Journal of Cyberpsychology and Behavior*, 2009

Peter J. Clarke, Jairo Pava, Debra Davis, Frank Hernandez, and Tariq M. King. A coupling-guided cluster analysis approach to reengineer the modularity of object-oriented systems. In *Using WReSTT in SE courses: an empirical study*, pages 30–7–312. SIGCSE, 2012

Yali Wu, Frank Hernandez, Peter J. Clarke, and Robert B. France. A dsml for coordinating user-centric communication services. In *Using WReSTT in SE courses: an empirical study*. COMPSAC11 (2011 35th Annual IEEE International Computer Software and Applications Conference), 2011

Christine Lisetti, Emmanuel Pozzo, Marie Lucas, Frank Hernandez. W. Silverman, and W. Kurtines. Programming in the second life virtual world. In *In Proceedings of the 14th Annual Conference on CyberTherapy and CyberPsychology Conference - Studies in Health Technology and Informatics (SHTI)*. IO Press: Amsterdam, 2009

Yali Wu, Andrew A. Allen, Frank Hernandez, Yingbo Wang, and Peter J. Clarke. A user-centric communication middleware for cvm. In *IASTED International Conference on Software Engineering and Applications*, pages 210–215. SEA, 2008

Yali Wu, Frank Hernandez, Francisco Ortega, Peter J. Clarke, and Robert France. Measuring the effort for creating and using domain-specific models. In *Proceedings of the 10th SPLASH Workshop on Domain-Speci?c Modeling*. DSM, 2010

Frank Hernandez and Peter J. Clarke. Towards integration of policies into dsmls. In *Proceedings of the 10th SPLASH Workshop on Domain-Specific Modeling*. DSM, 2011

Frank E. Hernandez and Francisco R. Ortega. Eberos gml2d: A graphical domain-specific language for modeling 2d video games. In *Proceedings of the 10th SPLASH Workshop on Domain-Speci?c Modeling*. DSM, 2010