



# **Universal BioSys**

*“Best security through nature”*

Submitted by

S. M. R. P. De Silva

H. M. N. D. Bandara

P. W. H. D. Weerasinghe

Department of Computer Science and Engineering

In partial fulfillment of the requirements for the Degree of  
Bachelor of Science in Engineering  
University of Moratuwa  
Sri Lanka



# Universal Biometric System

*“Best security through nature”*

Biometric enabled third-party authentication system

Final year undergraduate project

2000/2001 Batch

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

[www.biosys.net.tc](http://www.biosys.net.tc)



---

*Best security through nature....*

## Abstract

Biometrics is today's prime technology when it comes to access control, especially in medium to large scale organisations. At present the technology is matured enough and it has proven it is the current best when tight security is your main concern. It is convenient to use, publicly accepted and more importantly affordable. Users have all forms of biometrics technologies (Fingerprint, Iris, Retina, Facial, etc.) to choose from, based on the required level of security and available budget constraints.

So, why is it still not heavily deployed or what is the reason this is holding the market? It is the complexity of integration; most of the time it is too hard, too costly and in the worst case it is impractical. It does not easily fit into today's complex enterprise level networks. There are very few solutions that meet up this challenge, even those solutions are either limited to a particular biometric technology, vendor or platform.

Fortunately now we have a standard known as the BioAPI (proposed by BioAPI Consortium and could be consider the Defacto standard framework). However BioAPI is not directly deployable and user has to build an application on top of it to make it a deployable product.

Universal BioSys meets this challenger and it sets it sights far beyond the BioAPI. Universal BioSys is a third-party authentication system that hides all the complexities of biometrics and presents a simple development environment than the BioAPI. It offers considerable management and administrative advantage while considerably bring down the Total Cost of Ownership.

It introduces several unique features like in-depth security through Device-Hierarchy and seamless many-to-many mapping between applications and devices plus standard



network and security practises. Compared to other available products on the market BioSys is unmatched due to its broad scope and unique features.

Universal BioSys is primarily a Proof of Concept. It tries to prove that biometric integration can be made seamless and effortless while enforcing management and security practices. Universal BioSys team is proud to announce that the project was really successful and to prove such a solution is feasible. With enough time and effort, it could be turned in to a marketable product.

## Acknowledgement

It is with great pleasure the Universal BioSys team acknowledges the assistance, guidance and contributions (both direct and indirect) given by all the individuals who helped in making the project a success.

Special thanks goes to Mr. Chathura de Silva who was the project supervisor as well as the final year project coordinator for the valuable guidance and instructions. We must also thank Dr Sanath Jayasena the Head of the Computer Science and Engineering Department for allocating enough resources for project developments.

We would also like to thank members in The BioAPI Consortium who have been involved in the development of the specification and reference implementation.

The BioSys team would also like to thanks Mr and Mrs De Silva for providing accommodation at the integration stage of the project.

Finally we would like to thank our own batch members, system support team, non academic staff of the department who helped us in various ways and each and very individual that we have not mentioned above, who gave us the slightest support through the lifetime of the project.

*Thank you all, without your support there won't be a Universal BioSys....*

# Table of Contents

Abstract	i
Acknowledgement	iii
List of Figure	vii
List of Tables	ix
Keywords, Symbols and Abbreviations	x
<b>Chapter 1 – Introduction</b>	<b>1</b>
1.1 Previous Developments	3
1.2 Objective	5
1.3 Organization of the Thesis	6
<b>Chapter 2 – Literature Review</b>	<b>8</b>
2.1 Biometrics	8
2.2 Standards	9
2.2.1 Standards for Application Developers	10
2.2.2 Standards for Secure Communication and Financial Services	11
2.2.3 Standards for Exchanging Biometrics Data	12
2.3 Microsoft DirectX	13
2.4 OpenCV	13
2.5 RC2 Encryption	14
2.6 Socket Communication	15
2.7 Web Services	16
2.8 XML Parser	17
2.9 Database Support	17
<b>Chapter 3 – Our Solution</b>	<b>19</b>



3.1 Problem in Detail	19
3.1.1 Introduction	19
3.1.2 The Challenge	20
3.2 Extending the BioAPI	25
3.2.1 Introduction	25
3.2.2 Scope	26
3.2.3 The API Module	27
3.2.4 Distributed (Client/Server) BSP vs. Local BSP	28
3.2.5 BioAPI and Universal BioSys	30
3.2.6 BioAPI Wrapper	31
3.2.7 Architecture	32
3.2.8 Subcomponents of wrapper that interface the BioSys service	34
3.3 BioSysManager	35
3.3.1 Why BioSys Manager	35
3.3.2 Solutions Provided	36
3.3.3 Client Components	38
<b>Chapter 4 – Functionality and Implementation in Detail</b>	<b>40</b>
4.1 Architecture of Universal BioSys	40
4.1.1 BioSys Manager	40
4.1.2 Administrative Console	43
4.1.3 Application Client	45
4.1.4 Desktop Client	46
4.2 Cost Effective Face Recognition	46
4.2.1 Introduction	47
4.2.2 Key Features of The Facial Recognition System	48
4.2.3 Image Acquisition	48

4.2.4	Pre Processing	49
4.2.5	Eigen Face, Face Recognition	49
4.2.5.1	Face Recognition Algorithms	49
4.2.5.2	Dimensionality Reduction	50
4.2.5.3	PCA	50
4.2.5.4	History	51
4.2.5.5	Eigen Faces Algorithm	51
4.2.5.5.1	Eigen Face Calculation	52
4.2.5.5.2	Encoding of Faces	54
4.2.5.5.3	Recognition and Learning Phase	55
4.2.6	Services of the Facial Recognition System	55
4.2.6.1	Enrolment	55
4.2.6.2	Identification	56
4.2.7	Limitations	57
<b>Discussion</b>		58
<b>Conclusion</b>		61
<b>References</b>		65
<b>Appendix A – Development Process</b>		A-1
<b>Appendix B – Detailed Design</b>		B-1
	Use Case Diagram	B-1
	Activity Diagrams	B-2
	Database Structure	B-7
	Sequence Diagrams	B-8





# List of Figures

## Chapter 1 – Introduction

1.1	Sample biometrics architecture and the Independent Security Server	4
-----	--	---

## Chapter 2- Literature Review

2.1	Biometrics architecture and standards	10
-----	---------------------------------------	----

## Chapter 3 – Our Solution

3.1	Hypothetical floor arrangement for an organization	22
3.2	Device hierarchy for the layout given in figure 3.1	23
3.3	BioAPI as a layered model	26
3.4	Client/Server Implementation Using Primitive Functions	29
3.5	Client/Server Implementation using Streaming callbacks	30
3.6	BioSys components and their interactions	32
3.7	Sub layers of the BioAPI Wrapper	33
3.8	Subcomponents of the BioAPI Wrapper	34

## Chapter 4 - Functionality and Implementation in Detail

4.1	Basic components and their interrelationship	40
4.2	Main architecture of the BioSys	41
4.3	Components of a typical Image processing system	47
4.4	Vector representation of an image	52

## Appendix B – Detailed Design



B.1	Use Case diagram	B-1
B.2	Activity Diagram: User login	B-2
B.3	Activity Diagram: Add User	B-3
B.4	Activity Diagram: Update user	B-4
B.5	Activity Diagram: Delete user	B-5
B.6	Activity Diagram: Ban user	B-5
B.7	Activity Diagram: Policy enforcement	B-6
B.8	Design of the Database	B-7
	All the Sequence Diagrams	B-8

## List of Tables

*<there will be tables when performance test for the Facial Recognition System complete. So this page is reserved>*

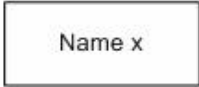
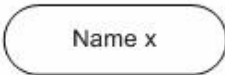

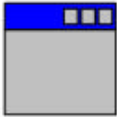





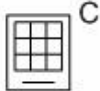

## Keywords, Symbols and Abbreviations

### Keywords

Biometrics	An open-ended set of technologies based on the measurement of some unique physical characteristics of an individual, for the purpose of identifying an individual (or verifying) identity.
BioAPI	An industry standard Application Program Interface (framework) defined by The Biometric Consortium.
Universal Biometric System	Refers to the entire solution and all its subcomponents
Universal BioSys	Is the product name that stands for Universal Biometric System.
BioSys server	The physical server that Universal BioSys runs on top of. It also includes the database server and BSPs.
BioSysManager	Is the web service that clients interact with is the core logic of the system. Also referred as the BioSys service.
BioSys Wrapper	The wrapper that wraps the BioAPI.



## Symbols

		Main or sub components
		Client applications
		User
		Biometric device
		Biometric Service Provider - BSP
		Web Cam
		Fingerprint scanner
		Card reader
		Database

## Abbreviations

ANSI	American National Standard Institute
API	Application Program Interface
ASN	Abstract Syntax Notation
BIR	Biometric Identification Record
BSP	Biometric Service Provider
CBEFF	Common Biometrics Exchange File Format
COM	Component Object Model
CSP	Cryptographic Service Provider
DLL	Dynamic Link Library
DNA	Deoxyribonucleic Acid
DOM	Document Object Model
FAR	Fault Acceptance Ratio
FRR	False Rejection Ratio
GUI	Graphical User Interface
HTTP	Hypertext transfer protocol
IIS	Internet Information Service
IPL	Intel Image Processing Library
IPP	Intel Performance Primitives
IPPI	Intel Performance Primitives Imaging
IPPSM	Intel Performance Primitives Small Matrices
IPPSP	Intel Performance Primitives Signal Processing
ISO	International Standard Organization
IT	Information Technology
MFC	Microsoft Foundation Classes
MMX	MultiMedia eXtensions
Open CV	Open Source Computer Vision Library
PDU	Protocol Data Unit
RFC	Request fro Comments
RPC	Remote Procedure Calls
SAX	Simple API for XML
SDK	Software Development Kit
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol

SPI	Service Provider Interface
SSE	Streaming SIMD Extensions
TCO	Total Cost of Ownership
TCP/IP	Transmission Control Protocol/Internet Protocol
UUID	Universal Unique IDentifier
WASP	Web Applications and Services Platform
WSDL	Web Service Description Language
WSS	Web Service Security
XCBF	XML Common Biometric Format
XML	eXtensive Markup language



# Chapter 1

## Introduction

Biometrics is an open-ended set of technologies based on the measurement of unique some physical characteristics of human beings for the purpose of identifying an individual or verifying identity. Simply saying your body is your password.

The importance, the value of Biometrics stem from three main factors. What you know (i.e., password or PIN) are insecure, can be forgotten, needs to be changed frequently, can easily be copied, given to others or could be guessed by others. What you have (i.e., ID card or a key) can be lost or copied (without your knowledge), and replacement cost could be high. And the most important and unique feature of biometrics is that it is the best non-reputable authentication method currently available.

There are many types of biometrics. Technologies like Iris/Retina are accurate, but invasive; hence Fingerprint/Hand Geometry/Facial approaches are widely used even if accuracy is somewhat compromised. Biometrics is not limited to the above-mentioned ones, but a long exhaustive list of technologies can be provided.

Despite the value proposition biometrics is not widely used. It is affordable, costs have fallen dramatically; there are many choices of proven technologies and it works well for the vast majority of users. Furthermore after the September 11<sup>th</sup> terrorist attack on World Trade centre there is a heightened sense of urgency to strengthen infrastructure security. So what has been holding the market back?

The big challenge or the bottleneck in biometrics is the integration. Biometrics has to become a fully integrated component within a complex enterprise level network environment and comply with other network based security practices.



Currently there are very few solutions that meet this challenge and those solutions are also tightly coupled to a specific vendor and to a selected set of technologies.

Universal BioSys is a third party identification (or verification) system that any application (in a networked environment) could use to identify (or verify) its users based on biometrics. The BioSys server detects and manages the biometric devices in the network relieving the applications from the technical complexities of the underlying biometrics. This solution would allow seamless many-to-many mapping between applications and biometric devices. The System is also concerned about various management policies, enforcement of such policies, in-depth security, near real-time monitoring, secure communication, adoption of industry standards and most importantly ease of management and administration.

Once the BioSys project members, decided to carry on with this intuitive and challenging project, a feasibility study was launched. Then we came to know of a standard prepared by the fore frontiers in biometrics and deemed to be the Defacto standard for biometrics.

The standard known as BioAPI [2] is the work of The BioAPI Consortium, which was formed solely to develop a widely available and accepted API to serve various biometric technologies. In a nutshell The BioAPI standard is a "framework", in which biometric software components ("Biometric Service Providers - BSP") are installed and advertise their capabilities by means of a standard registration mechanism, and the functionality they implement is made accessible to "biometric applications" via an application-programming interface.

It's important to note that, our objective is not to implement a specification as we frequently see a number of projects working around various RFCs. Universal BioSys tries to hide the complexity of biometric with in it self and provide a consistent, easy to use interface to the application developers and administrators. In doing so the product is based on BioAPI, as it will be the Defacto standard for biometrics. Hence BioAPI is



essential and core to the product, but it is not the product as in RFCs. For example the novel concept of Device-Hierarchy, which we have implemented, will be an eye opener for many security solution providers. Device-Hierarchy is a security policy particular to Biometrics. As an example a person cannot be authenticated to the server room facilities unless he has already been authenticated to the IT department premises. Hence the user may need to be authenticated through, several devices prior to authenticate through a particular device. Device hierarchy is directly related to departments and drag & drop based user interface should facilitate the construction of device hierarchy. This will also enable the administrator to keep track of user movement as well.

The architecture of Universal BioSys is designed to be scalable, so through the addition of various security related features, it can become a full pledged solution in the future.

## **1.1 Previous Developments**

A detailed study was carried out to determine the novelty of Universal BioSys and to understand the related industry. Our solution seemed to be unmatched. There were several security solutions that are related and worth mentioning.

### **The Independent Security Server – by Info Data, Inc.**

Provides means to identify a person based on any biometric characteristics (i.e. fingerprint, face, eye, palm, voice, or handwriting). They have developed BSP libraries for all the popular products therefore it is compatible with all major biometric scanners. Hence users have to rely on Info Data, Inc. [3] to provide compatibility with what ever the biometric device they buy. On the other hand Universal BioSys solution is very versatile. The customer can just plug-in any BioAPI complaint device and use without our intervention. Due to this Independent Security Server is not scalable or portable as Universal BioSys. Refer figure 1.1

### The WhoIsIt biometric server for E-commerce

This is basically an application server hosted in the Internet where a client system sends a biometric template to be verified. On success any secret (payload) that is stored for that

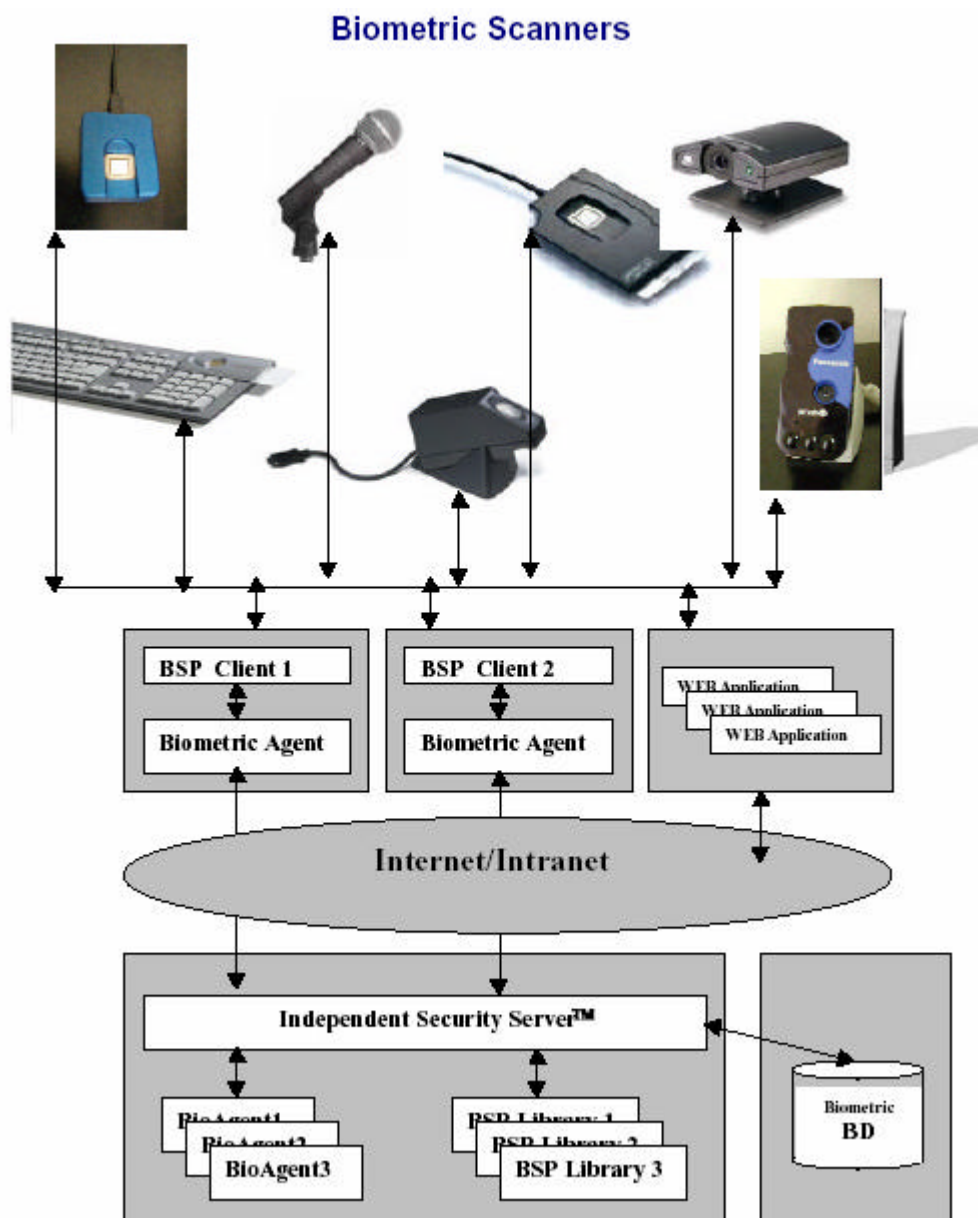


Figure 1.1 - Sample biometrics architecture and the Independent Security Server

particular user can be retrieved. An asymmetric cryptographic algorithm provides secure communication. WhoIsIt biometric server needs to be aware of the underlying technologies and the users are restricted to the vendors in commercial agreement with

them [4]. Therefore this is not as scalable or flexible as BioSys, but it has a quite mature design and implementation in deploying biometric services through the Internet. As Universal BioSys needs to be operational in any network, be it Internet or Intranet, a study of this was very useful.

## 1.2 Objectives

Universal BioSys is primarily a Proof of Concept effort. It tries to prove that biometric integration can be made seamless and effortless. The application developers no longer need to study vastly different APIs (i.e. SDKs) of biometrics devices. One fact needs to be emphasized here. Universal BioSys provides vendor independent and technology independent interface through the BioAPI standard. Then one might think BioAPI can be used to achieve the polymorphic behaviour (i.e. same function, but different implementations according to the vendor and technology) without Universal BioSys.

BioAPI consists of very complex data structures, the programmer needs to work with pointers of three levels of indirection (typically) and thereby need to undertake wearisome memory management. Universal BioSys hides all this dreadful, wearisome complexities and provides an easy to understand, and easy to develop interface to the application developers. Furthermore it is no longer a nightmare for the system administrator to manage different biometric devices of different vendors and different technologies. With this ambitious mindset four products are delivered.

The core is the BioSys server that is a complete implementation of the BioAPI version 1.10 framework. It runs as a web service, and coordinates applications requesting services and manages the biometric devices. Since the creators of BioAPI have developed a reference implementation of the BioAPI 1.10 framework and distribute it loyalty free, we tested it and developed a wrapper enclosing only the necessary functions in the context of BioSys. This was a hard task since the reference implementation was just a source code



(no necessary documentation were available) and the user support group is also current non-functional

Administration Console is a management console that can be kept either within the server or in a different machine. This is the place where parameter setting (i.e. FAR, FRR), construction of device hierarchy, user enrollment, and policy enforcement and near real-time monitoring is performed

The Biometric Client is typically a combination of: a physical device that extracts user data, a user interface, interfacing software (API), image capturing & image processing algorithms and a software component to communicate with the server. A fully functional Facial Recognition Client, plus several dummy simulated BSPs were developed to demonstrate the functionality of BioSys.

In addition to these three core components, BCB Generator is the other product in the Universal BioSys suite. BCB Generator (which stands for BioAPI Compliant BSP Generator) is an automated tool that generates fully BioAPI-compliant BSPs for non-standard biometric devices given specific device information. Due to time and human resource constraints it was limited to the specification.

## **Organization of the Thesis**

In Chapter 2, a summarised report of the literature study is presented. It is recommended to go through this chapter, as rest of the document refers its contents numerously. Chapter 3 is the core chapter. It starts by describing current bottleneck preventing the wide usage of biometrics. Then Universal BioSys is described as a solution that addresses this issue. Much of that chapter is devoted to describe the BioAPI standard and how it was wrapped elegantly to provide a vendor and technology independent biometric solution. In the latter part the chapter introduces the BioSys Manager; the web service that exposes the Biometric services.

The fourth chapter starts with a detail description of the functionalities provided by Universal BioSys. The second half is a concise description of the low cost facial recognition system we developed.

Then under the discussion, among other things prospective future directions are explained. The final chapter is a conclusion of the whole project. Then under the appendix we have annexed several documents through non-core, are worth looking at.

# Chapter 2

## Literature Review

Literature review was done mainly as a feasibility study to come up with a third-party identification and verification system that any application in a networked environment could use to identify (or verify) its users based on biometrics technologies. This review done under two main sections; previous developments related to the project and technologies that are suitable and planned to be used during the development and that are useful for the detailed design.

Rest of the chapter will concentrate on some of the basic technologies that were used during the implementation of BioSys. Those topics include: biometrics technology and its standards, Microsoft DirectX, Open CV, RC2 encryption, socket communication, web services and XML parser.

### 2.1 Biometrics

Biometrics is an open-ended set of technologies based on the measurement of some unique physical characteristics of an individual for the purpose of identifying an individual or verifying identity. Biometric is considered to be the most secure and convenient (public acceptance may differ based on cultures, physical inabilities) authentication technology.

The biometric is the most secure means of authentication that cannot be borrowed, stolen, or forgotten, and forging is practically impossible. This technology measures individual's unique physical or behavioural characteristics to recognize or authenticate their identity. Physical biometrics characteristics include fingerprint, hand geometry, retina, iris, and facial characteristics and behavioural characters include signature, voice, keystroke





pattern, and gait. Biometric has got the top attention for secure authentication methods for user verification and identification.

Biometrics can increase organization's ability to control access, protect its data by implementing a more secure key (referred as a payload) than a password and also it allows a hierarchical structure of data protection with using different type of biometric devices being employed within a single organization. Recent advancements in biometric sensors and matching algorithms have led to the deployment of biometric authentication in a large number of civilian applications. This technology can be used to prevent unauthorized access to physical and virtual areas and the day the biometrics become the mostly deployed authenticating mechanism is not that far.

## **2.2 Standards**

Any new technology undergoes lot of changes within a very short period of time after its introduction, resulting many products under various vendors (with variety of distinct and common features). After sometime there will be whole lot of standalone products or group of products that works together by the same vendor. However on the users' perspective, they would like to have mixed different vendor products with different features and that matches with their budget.

Lack of standards among products and variations of the technology will make it harder (could require lot of effort and money) or impossible to interoperate. Standardisation is the solution and bother users as well as the vendors could benefit from it. On users perspective they would get interoperable, better quality, vendor independence and more importantly it would reduce the cost of products due to the market competition. Vendors could benefit from increase in business and advanced technology but they will have to compete more.

This is no different to biometrics. Few years ago it was really messy and users were stuck into a single vendor, product, technology and platform. Integration is the biggest hurdle when it comes to biometrics and this is the main reason, which is holding is market.

However things have changed over time and there are several standards that are related to biometrics (figure x) and each standard operate at different levels of the biometric architecture. Fallowing sub sections introduce some key ones given in the figure 2.1.

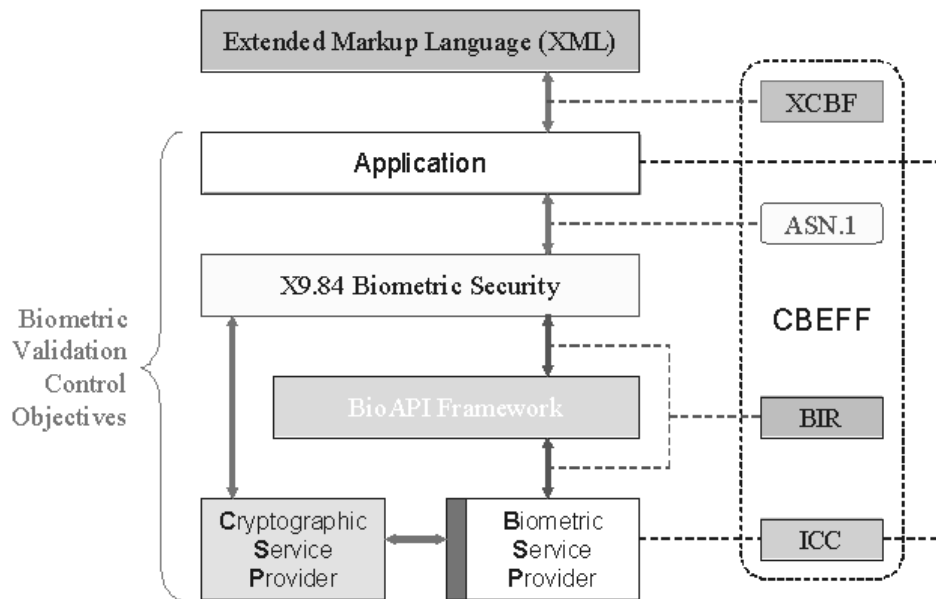


Figure 2.1 - Biometrics architecture and standards

Standards can be studied under the different layers and these layers represent different components of a biometric system. Any biometrics system can have all of the given components or part of it.

### 2.2.1 Standards for Application Developers

These are the standards provided by the developers of the biometrics device to the application developers. Most vendors tend to present their own SDKs to the application developers who want to make use of their products within user applications. Although the devices are affordable these SDKs cost a lot and developer needs to master new SDKs

when they move to a different vendor or even to a different technology given by the same vendor.

BioAPI is a standard developed by the BioAPI Consortium [2] (established in April 1998 by some of the frontiers in the biometric field and organizations like Intel) and this is the most primitive standard that any devices should support (BioAPI is accepted as an ANSI standard under ANSI INCITS 358-2002). BioAPI hides unique features of different technologies and vendors and provides a consistent interface to the application developer. BioAPI plays a key role in Universal BioSys, it can be considered as the core of BioSys. Section 2.2 discusses more about BioAPI in detail.

### **2.2.2 Standards for Secure Communication and Financial Services**

BioAPI only defines how a BSPs (Biometric Service Provider, which serves the requesting application) and applications should communicate; it does not define issues related to biometrics information management and security. A separate layer is added in-between the BSP and the application; to support secure communication and to accommodate applications that require tighter security, specially ones related to the financial industry and network based access control mechanisms. A separate sub-component called the Cryptographic Security Provider (CSP) is added to support both encryption and decryption.

X9.84 is one of the X9 standards defined by Accredited Standards Committee (accredited by ANSI). X9 develops and publishes voluntary, consensus technical standards for the financial services industry. X9.84 [6] standard defines, using the ASN.1 (Abstract Syntax Notation) language, a rich set of messages that are able to carry biometric data in a secure way. The standard also defines many concepts and procedures for the creation of a secure biometric system. The message formats specified by X9.84 are more flexible than the BioAPI data format (they allow a richer description of the biometric data that they carry,

and are extensible). Moreover, the X9.84 standard addresses the issues of integrity and privacy of biometric samples and templates, by providing several different security mechanisms among which the user can choose. The problem is that X9 standards are not freely available, however prewritten classes are provided with development environments such as Microsoft .Net.

### 2.2.3 Standards for Exchanging Biometrics Data

It is not just enough to securely communicate among vendors. There are requirements to use captured biometrics data among different systems that can be accessed globally. Such an example is; after September 11<sup>th</sup> attack, USA decided to capture and store all the fingerprints of foreigners that visit USA and that data was supposed to be retrieved anywhere within USA. Such a system could scale into universal biometric data storage.

The Common Biometrics Exchange File Format (CBEFF) was introduced to handle such issues. It defines things like; the length and width of the captured image, resolution of the scanner to be used, internal structure of the file, etc. CBEFF is also an ANSI standard.

Today XML plays a big role in representing both data and metadata, improving this idea further an XML standard called XML Common Biometric Format (XCBF) has been introduced specially for biometrics data that is designed to transfer through the Internet. XCBF is a common set of secure XML encoding for the formats specified in CBEFF. XCBF allows the use of biometrics in Web Services with the help of WSS (Web Service Security) specification.

Standards in biometric has really improved the market for biometric technology and currently ISO (International Standard Organization) is in the process of defining a global standard, that combines all the above mentioned and several other not so popular standards

### 2.3 Microsoft DirectX

Universal BioSys requires a Facial recognition system to demonstrate its functionality. Thereby it was necessary to use a graphics API to capture images and then save to the hard disk as a bitmap image.

Graphics APIs provides a standard platform that enables software developers to access specialized hardware (especially hardware related to graphics) features without having to write hardware-specific code. Microsoft DirectX, OpenGL and Glide are the three main Graphic APIs. Since the Facial Recognition System is Microsoft Windows based, at this stage and it relies on MFC support, we selected DirectX as the best alternative.

Microsoft DirectX is an advanced suite of multimedia APIs built into all the current Microsoft Windows® operating systems. DirectX debuted in 1995 and quickly became a recognized standard for multimedia application development on the Windows platform.

API controls a set of low-level functions that access the hardware or it provides hardware emulation if no hardware actually exists. These functions include support for 2D and 3D graphics acceleration, control over myriad input devices, functions for mixing and sampling audio and video output, control over networking and multiplayer gaming, and support for various multimedia streaming formats. Out of component APIs that handle these functions only DirectShow was relevant for our requirement.

### 2.4 OpenCV

OpenCV is the Open Source library of Intel Performance Libraries. The Intel Performance Libraries are a set of optimized C++ libraries providing simpler algorithmic development for scientific and mathematical applications, similar to the Matlab platform, but with the advantage of fully compiled (and speedy) C code.

These libraries take advantage of extended CPU features such as MMX and SSE. Thereby allow blocks of certain type of data to be processed simultaneously. Due to that, and to the performance of the C compiler, generated code runs much faster.

Several flavours of Intel Performance Libraries are available:

IPL	-	Intel Image Processing Library
IPP	-	Intel Performance Primitives
	IPPI -	IPP Imaging
	IPPSP -	IPP Signal Processing
	IPPSM -	IPP Small Matrices
OpenCV	-	Open Source Computer Vision Library

OpenCV contains a host of algorithms and samples for dealing with many computer vision problems. A useful feature is that while it can utilize IPP for better performance, it is also compatible with IPL. However it is an Open Source distribution and has had many contributions from various Computer Vision groups. Currently Beta release of version 3.1 is available.

The OpenCV core contains operations for contour extraction, line and ellipse fitting, local feature detection, Hough transform lines, local and masked statistics, pyramid formation, morphology, connected components, distance transform, camera calibration, image warping, optical flow, adaptive contours (snakes), kalman filtering, histogram analysis, eigen-vector analysis, hidden Markov models, simple matrix operations, and even gesture analysis and motion segmentation.

## 2.5 RC2 Encryption

Communication over public networks such as the Internet is susceptible to being read or even modified by unauthorized third parties. Cryptography tries to create secure channels of communication over otherwise insecure channels, ensuring

data integrity and authentication. Cryptography is accomplished through encryption algorithms and protocols. There are various classes of encryption algorithms, falling into two categories: symmetric key encryption and public key encryption.

RC2 is a conventional, symmetric key block encryption algorithm. The input and output block sizes are 64 bits each. The key size is variable, from one byte up to 128 bytes, although the current implementation uses eight bytes.

Microsoft CryptoAPI is a real innovative work of Microsoft to ease the development effort in writing cryptography applications in windows platforms (for their development languages, like Visual C++ and other .Net languages)

The classes in the Microsoft CryptoAPI manage many details of cryptography for you. You do not need to be an expert in cryptography to use these classes. When you create a new instance of one of the encryption algorithm classes, keys are auto-generated, and default properties are always as safe and secure as possible.

Therefore BioSys team used Microsoft CryptoAPI for secure communication. Out of the many encryptions algorithms Microsoft CryptoAPI makes available, RC2 was chosen due to its simplicity, less overhead and sufficient security.

## **2.6 Socket Communication**

Socket is an end point of communication and it is a set of transport primitive. API for sockets was originated in the UNIX environment and it was a generalization of file access mechanism. Currently all most all the network operating systems support sockets however its internal implementation may vary depending on the platform, still the socket primitive are the same.

A socket is an end point that bounds to a specific port number and address. Whatever the internal implementation sockets allow a means of communication between different platforms and languages. It just needs another socket at the destination.

Sockets are important in the context of Universal BioSys to communicate with remote biometric devices that will send user biometric data to the BioSys server. Socket communication is the most suitable means of communication with biometric devices through a TCP/IP based network since it is the standard that most devices would support (it would be unrealistic to expect that those devices would support high level mechanisms such as Web Services) and sockets can be written in C/C++ with considerable efficiency.

## 2.7 Web Services

Web service is a cutting edge technology, which performs remote method calls over HTTP using the SOAP (Simple Object Access Protocol). SOAP simplifies the communication overhead in generic remote procedure calls. SOAP messages are implemented on top of XML. A remote SOAP server, which is capable of understanding XML, based SOAP requests and responding with SOAP formatted responses to the client is called a web service.

A Web service is completely described using WSDL (Web Service Description Language) where WSDL provides the description for all methods that the client can request. WSDL is also based on XML and it provides the data types to be used during the communication of each method.

There are mainly three popular SOAP engines: Axis, WASP and Microsoft .Net web service. Axis is an Open Source SOAP engine, which was initially developed in C++, and currently Java version is also available. Axis is powered by Apache and it supports only Java development. Axis SOAP engine runs on top of Apache Tomcat server. WASP (Web Applications and Services Platform) is a platform-independent, commercial web



service product, which can be used for development in Java and C++. Microsoft .NET web service that comes with the Microsoft .NET framework which can be used to facilitate clients developed by Microsoft and java based development environments. Developments of the web service can either be done with C++ or C#. Microsoft .NET web service runs on top of Microsoft IIS (Internet Information Service) and it supports higher number of data types compared to other two web service engines [7].

Microsoft .NET web service provides security based on authentication and authorization where it is capable of executing requests based on client credentials. Since Microsoft .NET web service operates with Microsoft IIS SOAP based security can inherit features from Microsoft security solutions.

## 2.8 XML Parser

XML documents need to be processed inside a program to retrieve data from it. To process the document it has to 'parse'. A parser is a program that reads the XML file, confirms that the file is in the correct XML format, breaks the structure into constituent elements and finally let the programmer access to data in the document, based on the element names or positions in the tree structure. There are three kinds of XML parsers. The Document Object Model (DOM) parser reads an XML document into a tree structure where the Simple API for XML (SAX) parser generates events for elements or attributes as it reads the XML document [8]. Pull parser returns the data corresponding to a given element or attribute name. The Microsoft .NET has a Sax parser, which comes with the class, name XMLTextReader, which is mostly used in XML parsing.

## 2.9 Database Support

Biometric data involves various templates as well as raw images so that the database server should be capable of handling such huge capacity of data. Ability to store images

in the database and less development overhead when connecting to the database are critical factors in selecting a database system. Oracle, SQL Server, Postgress and MySQL are some of the leading products. Postgress and MySQL are open source databases, which are highly, recommended for Linux platform. Both Oracle and SQL Server supports the Microsoft Windows platform and they satisfy the requirements of Universal BioSys.

Universal BioSys team selected Microsoft SQL Server 2000 since it experiences lesser development overhead with .NET environment, as it includes inbuilt classes that support SQL Server. It also facilitates the requirement of storing images to the database and the use of stored procedures. Stored procedures improve performance by reducing the time to execute a given query since it is already compiled with in the database. SQL server 2000 was the best option since it also supports distributed databases as well.

All of the above mentioned technologies were used during the implementation of BioSys. Several technologies like VPN that were included with the initial literature review were leftover since scope was too much.

## Chapter 3

### Our Solution

Universal BioSys is designed to overcome some of the basic problems relating to biometrics, the integration and it also offers several distinct features. Rest of this chapter concentrate on the problem that BioSys is tackling and its approach towards it. This also concentrates some of its unique components the BioAPI Wrapper and BioSysManager.

#### 3.1 Problem in Detail

Before getting directly into the exact problem it would be better to have bit of background knowledge about biometric and related practices.

##### 3.1.1 Introduction

Biometrics is the number one option when it comes to access control in today's medium to large-scale organizations. Biometrics is popular because of three main reasons: it replaces the requirement of easily forgotten and easily guessable passwords and also bulky magnetic, barcodes and crypto cards.

Second reason is who ever the user he/she has to be physically present in order to access something. It is a common case that people give their magnetic or crypto cards (or even passwords) to other to do things that they should do. This is a common problem in garment factories in Sri Lanka (where workers give their punch cards to others so that they can prove that they came on time) and several factories have moved into fingerprint enabled access control systems to keep track of time and attendance.



The third reason is, it is more secure since it is hard to forge a biometric system. The level of security depends on the technology being used where face recognition and hand geometry systems are less secure compared to fingerprint and retina based systems.

Therefore biometrics is secure and convenient but as every thing else in this world it has its own disadvantages. Higher cost of ownership, difficult to use, and lesser public acceptance are some of the problems faced by the biometric technology. Over time costs have fallen dramatically, there are whole lot of technologies to choose from that can work with large number of users, etc.

At present, biometric technology is matured enough, but the biggest hurdle is the integration among different technologies, vendors and platforms within a complex enterprise level network. However biometrics is the future and there are very few solutions that can meet the challenge. Universal BioSys is developed to prove that this hurdle can be beaten with economical and administrative advantages.

### **3.1.2 The Challenge**

When your business grows your IT infrastructure and other related technologies need to grow with you. So your technology framework should be scalable enough. It needs to grow from few machines from one or two vendors in one platform to a hundreds to thousands of machine from whole lot of vendors in different platforms. In here the challenge is the interoperability among different vendors and platforms. This is same with your access control (or if we call the security) system. From time to time you will purchase different devices (can either be card readers or other forms of biometric) from variety of vendors. Those vendors may not exist after few years or the product may be out dated.

Things work fine until you try to integrate and manage them centrally. Biometrics being a relatively new technology it lacks standards. So now you are stuck with the problem of platform and vendor dependence.

Applications that make use of biometrics are designed for particular type of biometric device of a specific vendor. In order to develop such applications they need to understand the internal behavior of the particular device or need to master the SDK given by the vendor. This SDK varies from vendor to vendor and some times from biometric technology to another (as an example; it may be different SDK for fingerprint and another for voice recognition system). Applications have to be rewritten even when biometric technology or vendor changes. This will require development to learn several SDKs or to stick to particular technology and vendor. This problem will also limit the use of platform independent languages for application development since SDKs are platform dependent. For such applications both the development and maintenance cost can be quite significant.

Now we are in to three problems:

1. Vendor dependence
2. Platform dependence
3. Technology dependence

Non standard compliance and technology dependency in biometric make it impossible to combine several type of devices together. Organizations cannot accommodate same type of biometric devices all over the organization. They would prefer to have low cost devices at places where threat is potentially lower and prefer high-tech (also higher cost) devices only when threats are high.

Most of the organizations are in a dilemma that they have to scrap all their card readers while moving to another biometrics technology such as fingerprint systems. They will

face the same problem again when they wish to move in to retina, iris or DNA when they are affordable.

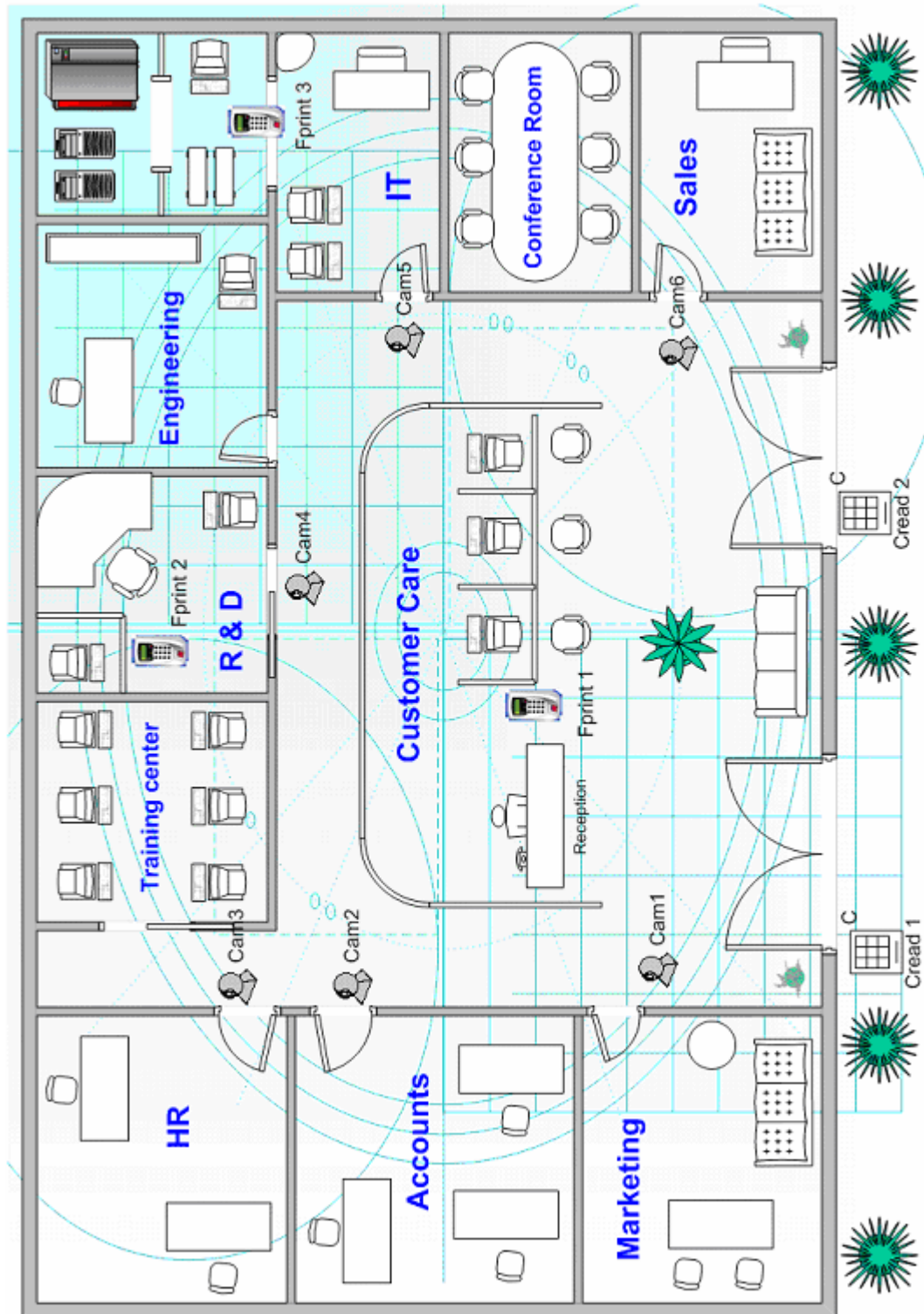


Figure 3.1 – Hypothetical floor arrangement for an organization.

It will certainly raise few eye brows in the board meeting if IT manager or security officer prepares a proposal asking to install fingerprint systems at each and every door step and PC. What they require into integrate different biometric and non-biometric technologies so that they neither invest too much or too les. They require a balance in their TCO (Total Cost of Ownership).

Refer figure 3.1, such an arrangement of devices creates a hierarchy of their own (see figure 3.2) that can be exploited to allow in-depth security. Such a device arrangement would reduce the TCO while providing appropriate level of security. Particular user has to go through several devices when they get into different departments within the building. Users may by pass this sequence either deliberate or accidentally in order to gain access. In cases where security is the prime concern such behaviours should not be allowed. There is a requirement to enforce such access rules for better security. It will prevent some one coming from the roof login in to a server in the server room. Following such an approach will indicate the location of the user based on his/her last authentication point and this is essential in organizations that span several square kilometres or on a building with multiple floors.

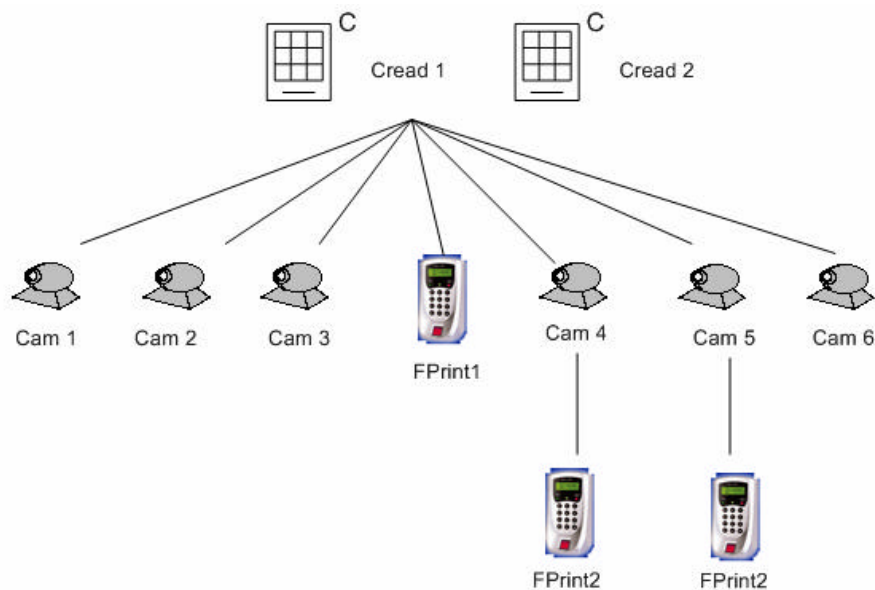


Figure 3.2 – Device hierarchy for the layout given in figure 3.1

Current biometric approaches require each host or server to have a dedicated device of its own. So if you have hundred machines you need hundred devices plus what ever number of devices that you install at your doors, this is not feasible. Managing such a large number of devices is not an easy task and the budget is a bigger concern. So there is a requirement to reduce the total number of required biometric devices used within an organization. If at least two hosts can share one device, total cost would reduce by half, hopefully with lesser administrative effort. This is a big economic and administrative advantage for large software development organizations, computer labs in campuses, etc.

We may have the latest technology on hand but backward compatibility with conventional passwords and card readers is still required. So it is essential to support those as well, due to their wide spread usage. When things are integrated and managed through a network that task will be given to the IT team. And this will be another network related administrative tool from the point of view of the system administrator. So it is essential that such a solution adheres to common network standards, interfaces, functionalities, etc. even though it deals with biometrics.

Biometrics is supposed to provide security and if it is through a network security becomes a major concern. Any security tool must follow at least the standard practices. Users BIRs (Biometric Identification Records) that travels through the network should be at least signed, if possible encrypted, passwords should also be encrypted, the solution should be robust to withstand buffer overflow attacks, etc.

There is a huge interest to convert non standard biometric devices so that they adhere to standards and could be integrated with others. So any solution that tries to solve all the above mentioned problems have to tackle this problem as well.

If we just summarise various major and minor problems that needs some solution:

- Interoperability among vendors, platforms and biometric technologies.
- Complex and non scalable application development environments



- Higher TCO associated with using same type of biometric technology
- Requirement to use shared devices for multiple hosts
- In-depth security through better utilization of available devices
- Backward compatibility with password and card readers
- Development of a tool that fits into a complex network environment, that allows centralized management, administrative advantage, etc.

## **3.2 Extending the BioAPI**

BioAPI is an Application Programming Interface (API) that is intended to provide a high-level generic biometric authentication module; one suited for any form of biometric technology. This section concentrates on the BioAPI, its pros and cons, relationship between BioAPI and Universal BioSys, BioAPI Wrapper and its sub components.

### **3.2.1 Introduction**

The BioAPI Consortium [2] was formed in 1998 to develop a widely available and accepted API to serve various biometric technologies. The outcome is The BioAPI specification/standard.

In a nutshell The BioAPI standard defines, in complete technical details a "framework", in which biometric software components ("Biometric Service Providers - BSP") are installed and advertise their capabilities by means of a standard registration mechanism (the module registry), and the functionality they implement is made accessible to biometric applications via an API. The standardization of biometric software as BioAPI compliant BSP will undoubtedly lead to the emergence of a market of plug-and-play biometric components.

It covers the basic functions of Enrollment, Verification, and Identification, and includes a database interface to allow a BSP to manage the Identification population for optimum

performance. It also provides primitives which allow the applications to capture samples (user biometric data) on a client, and the Enrollment, Verification, and Identification to be performed on a remote server.

### 3.2.2 Scope

The BioAPI is two fold:

- Application Programming Interface (API)
- Service Provider Interface (SPI)

#### Application Programming Interface (API)

This is also called the *BioAPI h\_layer* (or High Level layer) and it provides all the functions that an application needs in order to perform biometrics authentication. Specification tries to hide as much as unique characteristics of individual biometrics technologies, vendor implementations, and products and provides high level abstraction to the application developer.

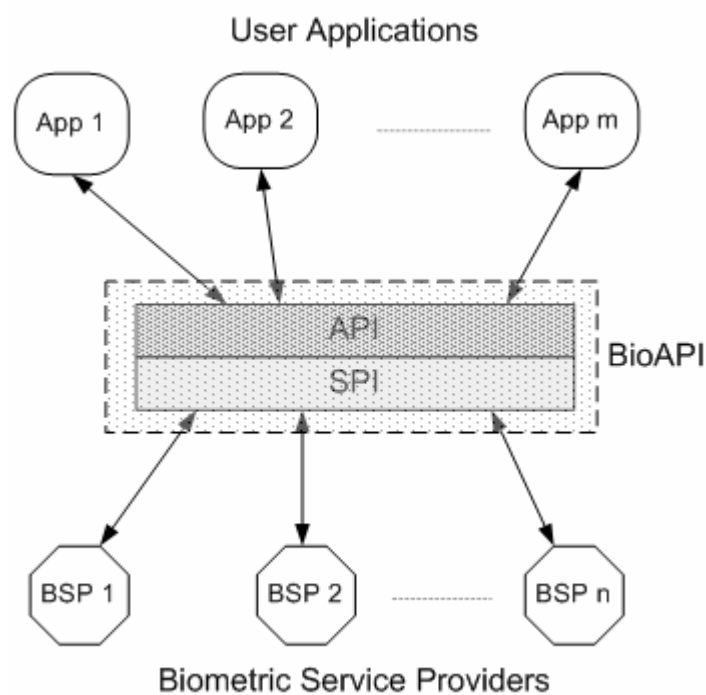


Figure 3.3 - BioAPI as a layered model

Therefore, to the extent possible, the amount of optional functionality is kept to a minimum. The major optional function is Identify; and the database capability is also optional and is defined in the interface primarily to allow the BSP to manage large populations that are used for identification.

The API does not address security requirements for biometric applications and service providers but recommendations are provided, on using the API to support for good security practices.

### **Service Provider Interface (SPI)**

The service provider Interface (SPI) is the programming interface that a BSP must manifest in order to plug into the BioAPI framework. SPI is almost a one-to-one mapping of the BioAPI down to the BSP. The framework routes API calls down to the corresponding SPI of the attached BSP. Not all API functions have a corresponding SPI function; and are handled by the framework (functions related to module registry management).

### **3.2.3 The API Module**

An application can use make use of biometric services in two fundamental ways:

#### **Primitive functions**

Primitive functions are the most basic functions that a BSP should have internally. These functions are the four primitive functions: Capture, Process, Match, and Create Template, that we encounter with any form of biometric technology.

The fact that the functionality of these functions vary from one BSP to another introduces an element of unpredictability. Furthermore the portability of an application from one BSP to another is also questionable. For applications to make use of these primitive

functions, the BSPs need to implement them. Then these functions of the BSP can be routed through the API and made available to the application.

Hence a lot of thought was given to making primitive functions compulsory for a BSP. It was decided that undue burden on self-contained devices (where biometric processing/matching is performed within the device itself) manufacturers is not justifiable. Therefore, these functions are not considered compulsory for BSP's to be "BioAPI compliant". However, if the underlying technology supports, the BSP should try to include those.

The solution is to use set of abstract functions and there are three principal high-level abstraction functions defined in the BioAPI:

- 1) Enroll        Samples are captured from a device, processed into a usable form, from which a template is constructed, and returned to the application.
- 2) Verify        One or more samples are captured, processed into a usable form, and then matched against an input template. The results of the comparison are returned.
- 3) Identify       One or more samples are captured, processed into a usable form, and matched against a set of templates. A list is returned showing how close the samples compare against the top candidates in the set.

These three functions are made mandatory for each BSP. Therefore only the BSP needs to be aware of the capabilities of the underlying technology/device. It can use a combination of the capturing processing or matching, which are irrelevant to the BioAPI as each BSP abstracts those complexities and provides enrol, verify and identify functions.

### 3.2.4 Distributed (Client/Server) BSP vs. Local BSP

The API offers biometric developer the maximum freedom in the placement of the processing involved, and allows the processing to be shared between the client machine (where the biometric device is attached), and on a server. Such is scenario is called a

Distributed (or Client/Server) BSP. Distributed BSPs allow: an environment where algorithms can execute securely, it reduces the require level of processing power on the client and it also offers centrally managed and securely stored user data.

Local BSP is the standard and simple scenario where the client application, the BSP and the biometric device all reside with in the same machine. Applications can request biometric services through the primitive functions (if supported) or the abstract functions directly from the local BSP.

Client/Server BSP implementation has two approaches: one approach makes use of primitive functions while the second approach use streaming callbacks. Figure 3.4 indicates the communication model with primitive functions. In here the requesting application is responsible for sequencing and synchronizing the client/server communication.

A callback function is a communication means, an application offers to BSPs and with Streaming callbacks the service provider can stream data to the application in the form of a sequence of protocol data units (PDU). It is the responsibility of the application to provide a streaming interface for the client and the server BSP. The three abstract functions make use of the streaming interface to split the BSP functions between the client and server. These functions can be initiated from either the client or the server.

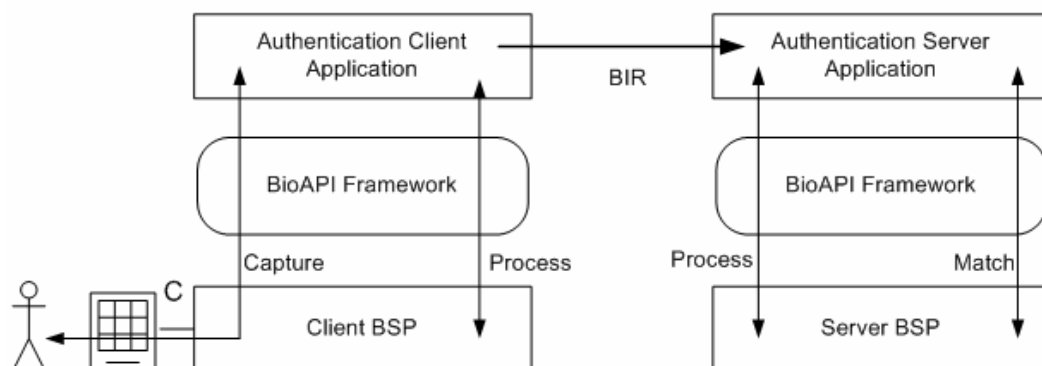


Figure 3.4 - Client/Server Implementation Using Primitive Functions

The application calls the appropriate high-level function, and then the BSP calls the streaming callback to initiate the BSP-to-BSP protocol. (The protocol is the concern of the BSP). The streaming callback is only used by the driving BSP. Whenever it is in control, and has a message to deliver to its partner BSP, it calls the streaming callback interface to send the message, and it receives an answer on return from the callback.

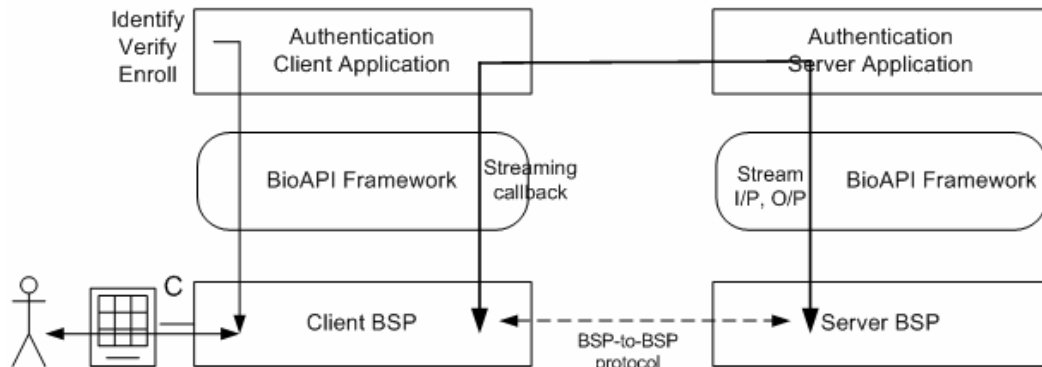


Figure 3.5 - Client/Server Implementation using Streaming callbacks. Operation is initiated by client

The Stream Input/Output function is used by the partner application to deliver messages to the partner BSP, and to obtain a return message to send to the driving BSP. The driving application delivers the return message by returning from the streaming callback.

### 3.2.5 BioAPI and Universal BioSys

Universal BioSys offer biometric technology, vendor and platform independency through some of the features addressed by the BioAPI. Therefore BioAPI can be considered as the core of Universal BioSys.

Although reference implementation of BioAPI is freely available no one can directly deploy it into working system an application should be developed on top of it. Universal BioSys is such an integrated solution that is deployable and customizable so that it fit into requirement of the medium to large scale organizations.

Aforesaid features are accomplished by API and SPI layers of the BioAPI specification. BioSys use a combination of primitive and abstract functions where abstract functions are used by the BioSys service layer (also referred as the BioSys Manager) while primitive functions are used to request services from BSPs. BioSys make use of the distributed BSP approach with primitive functions where all the processing is done at the BioSys server.

Current specification of BioAPI does not directly fits into the framework of Universal BioSys so another in-between layers (which we call the *BioAPI Wrapper*) is added by BioSys developers to fill in the gap.

### 3.2.6 BioAPI Wrapper

The main function of the BioAPI Wrapper is to transfer between abstract and primitive functions, while providing simple interface to the BioSys Service. According to BioAPI reference implementation developer needs a whole lot of other function even when calling a single abstract function. The wrapper is developed from scratch by BioSys develops and it offers fallowing advantages:

- BioAPI wrapper is a collection of classes that simply fits into the standard Object Oriented approach, so that the developer does require only initiating an object of the wrapper then he/she can call any of the abstract functions.
- It hides all the complexities of data structures, pointers and memory management that BioAPI essentially require. While doing so it will enable all the required functionality implemented in BioAPI and compiled into DLLs (in Windows platform only).
- Web service developer does not require through understanding the functions defined in BioAPI. They can concentrate only on the abstract functions (not on the other function that needs to be called before or after an abstract function).

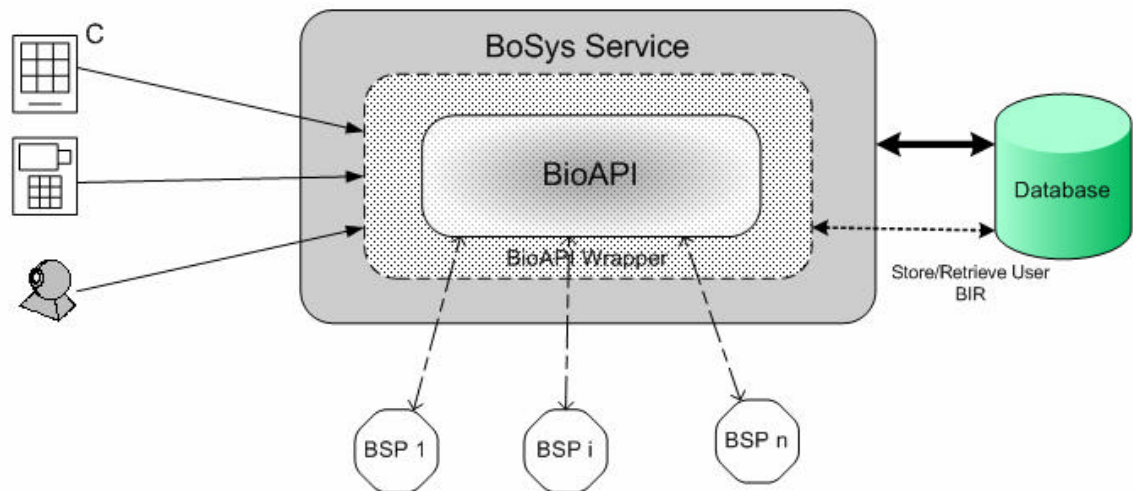


Figure 3.6 - BioSys components and their interactions. Arrows indicate parties involved in each communication endpoint and direction of communication.

Figure 3.6 illustrates the wrapper, its related components, communication among those components and the layers involved in communication. As indicated in the figure communication can bypass different layers.

One thing to note is that current implementation of BioAPI Wrapper does not wrap all the functions in BioAPI and it only deals with functions that are required in the context of BioSys. This wrapper is scalable so that it can be further extended, if more functionality is required in future.

The wrapper also handles other tasks like storing and retrieval of user BIRs (Biometric Identification Records) to and from a central database. It also listens to BIRs send by remote clients (i.e. devices) through a socket interface. We get into more detailed explanations of these services in coming subsections.

### 3.2.7 Architecture

Figure 3.7 indicates different layers in BioSys and notice that BioAPI Wrapper resides in between the BioSys Service and the BioAPI. BioAPI Wrapper is twofold: part of it lies at the API (i.e. BioAPI h\_layer) and interacts with the BioAPI. Although this is logically a different sub layer, it is written into the same source code as the BioAPI and compiled



into the same DLL. Other portion of the Wrapper is a collection of C# classes which BioSys Service talks into. When these two sub layers are combined together we called it the BioAPI Wrapper.

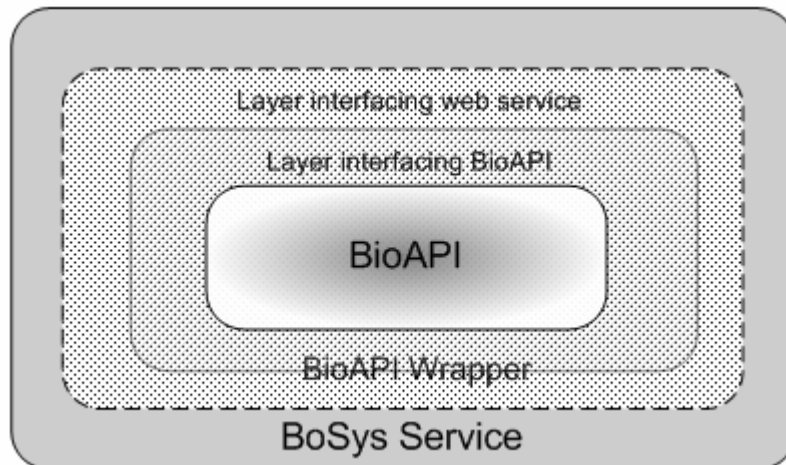


Figure 3.7 - Sub layers of the BioAPI Wrapper

Wrapped functions written in the sub-layer interfacing the BioAPI is accessible only through .Net interoperability services (System.Runtime.InteropServices). This is the mechanism that .Net use to request services from legacy DLLs and COMs that essentially require marshalling of pointers. Still this functionality is somewhat limited that is the reason why the BioSys developers had to add a sub layer interfacing the API (i.e. BioAPI h\_layer). So this is an indirect wrapping approach.

Layer interfacing the BioAPI uses primitive functions while layer interfacing BioSys Service uses abstract functions.

Layer interfacing the BioSys Service also communication with remote biometric devices, store and retrieve BIRs other than just wrapping the layer interfacing the BioAPI. These tasks are accomplished through several subcomponents within the wrapper.

### 3.2.8 Subcomponents of wrapper that interface the BioSys service

Aforesaid subcomponents in the layer interfacing BioSys Service are responsible for tasks other than mapping from abstract to primitive functions. These subcomponents are indicated in figure 3.8, it also indicates the various steps involve when enrolling a user.

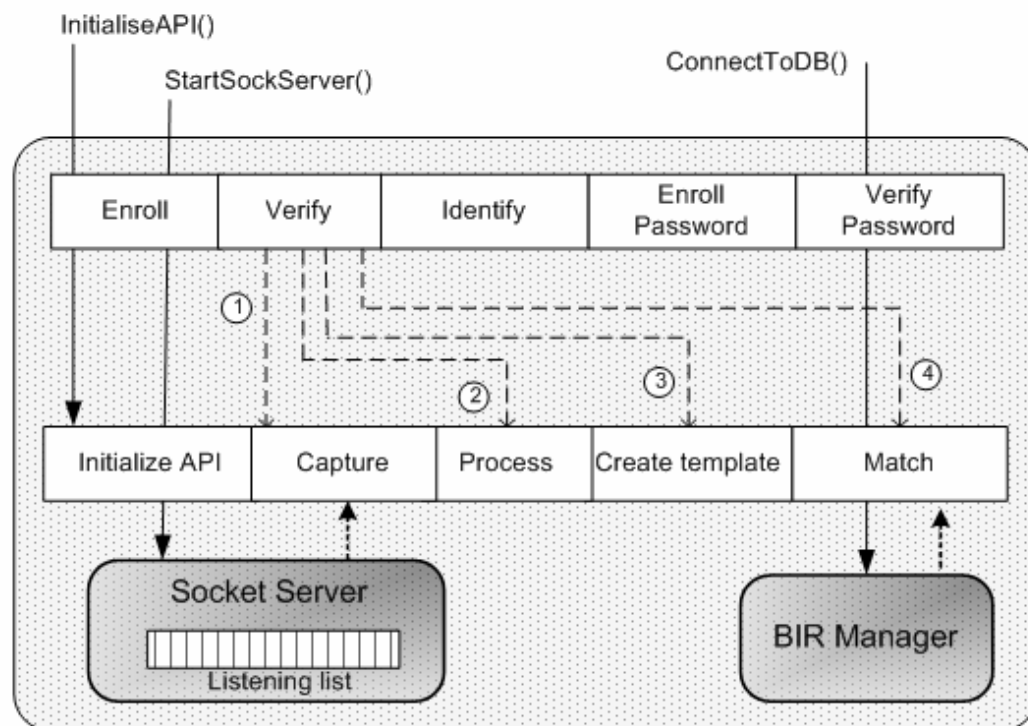


Figure 3.8 - Subcomponents of the BioAPI Wrapper

Five boxes at the top indicate the three abstract function plus two overloaded methods that handle passwords related functions. Next five boxes indicate the four primitive functions plus a special Initialize API function that initialize the BioAPI and its module registry.

Socket Server serves the capture request from the capture primitive function (this is distributed BSP approach with primitive functions). It makes use of network sockets. When capture request is given that request is stored in a link list (given as Listening list in figure 3.8) and server waits until it receives a BIR. When received it checks with the Listening list and send back to the capture function.

BIR manager is responsible for storing and retrieval of BIRs in a central database and it is used by match function.

InitialiseAPI(), StartSockServer() and ConectToDB() methods will initiate the BioAPI, load the module registry, start the Socket Server and get connect to BIR database through BIR manager. After that developer can call any of the abstract functions as he/she wish.

Each of these abstract functions includes several or all primitive functions. In figure 3.8 we can see the primitive functions related to Verify function and order of primitive functions calls are indicated by labelled numbers. All such activities and internal communication is hidden within the BioAPI Wrapper.

### 3.3 BioSysManager

BioSysManager is the key component of the Universal BioSys. It includes the core logic that enforces policies and also it presents a simple development environment for BioSys client applications. BioSysManager is a Microsoft .NET web service which serves different forms of clients with platform and language independency through a simple development environment. It is developed using the .Net C# and deployed on top of Microsoft IIS. BioSysManager also handles all the database operations related to users, biometric devices, hosts and policies.

#### 3.3.1 Why BioSysManager

BioSysManager is implemented using cutting edge technologies which include web services that enable RPC over HTTP using the SOAP. The use of SOAP engine reduces both the server/client development/deployment overheads involved in generic RPC programming. Clients can be developed and deployed using any technology however their developments have to be based on SOAP. So BioSysManager is capable of serving different and large amount of client applications which run on different platforms.



### **3.3.2 Solutions Provided**

Universal BioSys addresses the main issues of integrating biometrics within an enterprise level network. It offers technology, vendor and platform independency plus easy to use development environment (through web services) that client application developers will be benefited.

BioSys will enable different device types (such as card readers, fingerprint scanners, face recognizers, etc) to be installed within an organization and centralised management of those devices through an administrative Console.

BioSysManager keeps track of different device types, installed devices, location of those devices and the logical hierarchy that they produce. Administrator has to indicate the location (by means of a floor layout of the organization) of the device when it is registered in to the Universal BioSys. Location information about devices and hosts will lead to a better decision in selecting the most suitable device for a particular user to submit his./her biometric data. To gather the location information floor layouts of the organization are provided to the users which are saved in the system during the floors are added.

Another main entity of the BioSysManager is the user applications where it keeps record of all the applications that make use of Universal BioSys, in the organizations where the user authentication is required. These applications can be developed and deployed in any programming language which supports SOAP. Languages such as Java, C#, C++ and PHP already supports SOAP while languages like Perl, has certain SOAP plug-ins available. Therefore BioSysManager can provide its service to vast number of applications (even application through Internet) developed and deployed with different technologies.



BioSysManager also supports password authentication other than biometric authentication. This approach has two advantages: it makes the system backward compatible and it also enables remote login through the Internet. Home workers and remote users can log into online applications through internet using password authentication rather than physically be at the office to authenticate him/her self with his/her biometric data.

When devices are added to the system administrator has the ability to add those in to the Device-Hierarchy and this allows the administrator to enforce those users to follow the device hierarchy which allows in-depth security. See figure 3.2. BioSysManager make sure that users are unable to bypass the hierarchy and if they try to do so it will be logged and alert is send to the administrator.

Let's consider hypothetical example for the organization given in figure 3.1; if a user wants to access Server X in the server room he/she has to first authenticate form the card reader at the entrance, then the face recognition system at point of entrance to the IT department. Then in order to go in to the server room he/she should authenticated through fingerprint scanner and finally in order to login to the Server X he/she has to be authenticated from an Iris scanner (not included in figure 3.1). So no user can straightway access the Server X ignoring or by passing the rest of the device in the hierarchy. This is the one of the major and unique feature that come handy and it is in built in the Universal BioSys.

Hosts within the organization are needed to be registered into BioSysManager and then each host gets a unique identification (i.e. UUID). The location of the machine is also maintained in the system with respect to the organizations floor arrangement.

Employees in the organization have to be enrolled to the system providing their personal and contact details, and then they can be enrolled to respective applications that the particular user will be allowed to use. For each application particular user can select more



than one device type to be authenticated including passwords. Then the user is asked to submit his biometric data to a particular device and it gets captured by the BioAPI Wrapper [section x] and saved in a database. Username for each application and device type are maintained for verification purpose. BioSysManager provides the verification and also the identification functionalities to its clients.

For enrolment, verification and identification user has to go to a particular device and submit his /her biometric record. BioSysManager is intelligent enough to inform the nearest and, unoccupied device to the user according to his/she current location within the organization. For example when a user wants to log in to a particular application form a machine by using the fingerprint authentication the BioSysManger selects the closest fingerprint device to the particular machine and informs the user the device location (indicated by map of the organization) and the device name.

The service side can be more flexible to work even with a distributed database system as a result BioSysManager can even be deployed into organizations with remote branches through distributed databases. The BioSysManager provides all the main functionalities required for a generic biometric security system and it is scalable enough to add more functionalities if required. Universal BioSys is designed to facilitate requirements of a medium to large scale organization with large number of users, applications and hosts; however there is nothing that limits its use in a small scale organization.

### **3.3.3 Client Components**

Three client components communicate with the BioSysManager such as: the Administrative Console, Application Client and Desktop Client for various purposes. All these clients are developed using the .Net C# however Application Client and Desktop Client functionalities can be implemented using any other SOAP supported programming languages.



Administrative Console is a separate management Console that allow the administrator to manage, monitor and administrator activates and set various server parameters. These tasks include addition, updating and removal of device types, devices, hosts, users and near real-time monitoring, searching for users, etc. Administrative Console supports all the major administrative tasks should be carried out in an organizational network environment.

Application clients can request for verification and identification services from the BioSysManager. This can be done through a COM (Component Object Model) component in the Microsoft Windows platform for products develop using the language such as: Visual Basic, Visual C++ and other .Net based languages. The application developer can use the COM component in his application and call the given methods to verify and identify users.

Desktop Client application is installed in each host registered with the Universal BioSys. Through the desktop client each host gets the UUID and it shows all the loggings that are made to the different applications from that particular host. Users can easily logout from their applications using the desktop client.

The Administrative Console, Application Client and Desktop Client can be installed on Microsoft Windows machines with a customised Installation wizard.

## Chapter 4

### Functionalities and Implementation in Detail

#### 4.1 Architecture of Universal BioSys

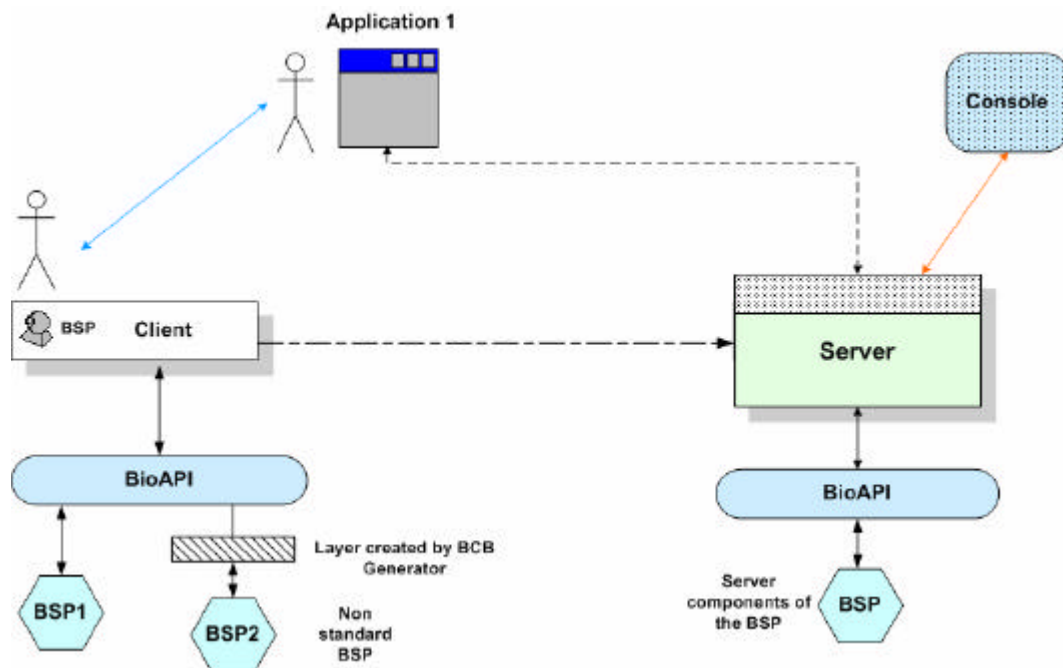


Figure 4.1 – Basic components and their interrelationship

Universal BioSys consists of several components such as: The BioSysManager, Administrative Console, Application Client and Desktop Client. Figure 4.1 illustrates different components and their interrelationship. BioSysManager is the server of the system which serves all three forms of remote client types. Server and client communication is based on HTTP. Biometric devices communicate directly with the BioAPI Wrapper [Section 3.2.6] through a socket connection.

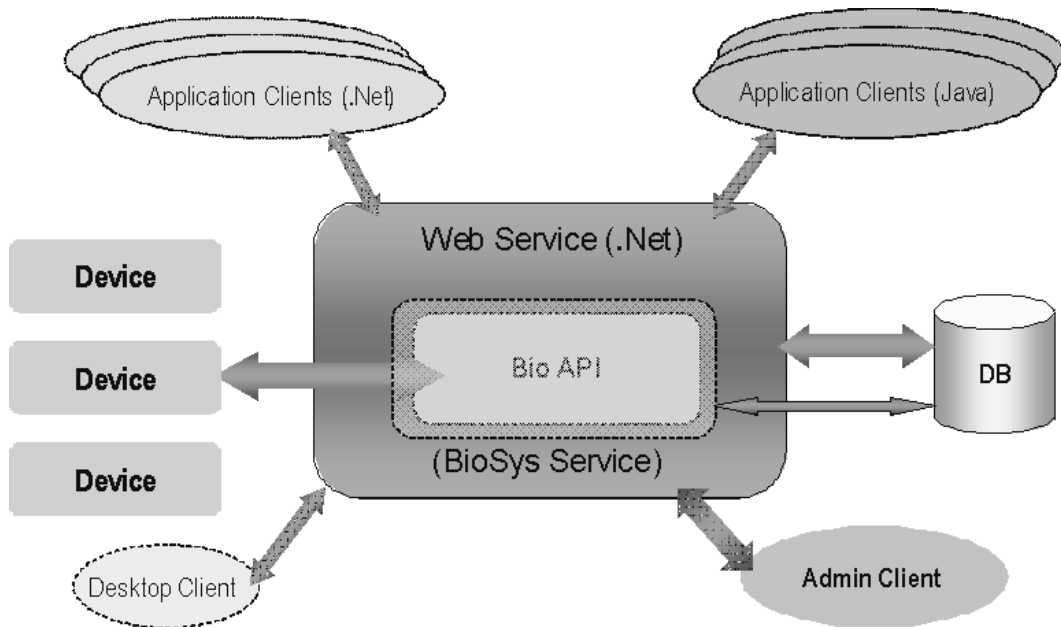
##### 4.1.1 BioSysManager

BioSysManager is implemented on basis of Object Oriented Architecture however the web service interface which is the main class of the BioSysManager is more towards the





SOA (Service Oriented Architecture). With SOA approach it makes possible to serve different types of clients with various services through a single interface. Anyway the internal implementation was fully followed with Object Oriented concepts.



4.2 – Main architecture of the BioSys

BioSysManager is developed in .Net C#, C and C++ languages and it is deployed in Microsoft IIS. Therefore BioSysManager can be accessed by local as well as remote clients.

BioSysManager contains classes for each entity that can be identified in the organizational environment, related security and network practices for a network based system. For example User, Device, Machine and Device-Hierarchy are some of the classes that can be found in the module. Most of the major entity classes implements four types of interfaces: ISavable, IViewable, IRemovable and IEditable. Add, Get, Remove and Edit methods in those classes are called through these four interfaces. BioSysManager main class has only four methods named Add, Edit, Remove and Get to carry out those functions for all the objects in the system. Inside each method in the main class, the particular object casts to the relevant interface and call the method in the interface. With this approach remote methods that had to be used in the BioSysManager

have been reduced rapidly otherwise each object type distinct Add, Remove, Edit and Get methods needed to be implemented. Wrapper classes are used to pass object arrays to client side and also to prevent serialisation errors when sending non-serialable data types directly from the BioSysManager to the client side.

If a particular remote method does not have a specific return object to be sent to the client side service it always sends an Error object which has an error number and an error message. If the method execution is successful, error value is set to 1 (Successful) else value is set to a constant value with specific definitions such as Failed, Login Incorrect and Already Exists.

Device-Hierarchy (implemented through a tree data structure) is required to make sure that users are not bypassing organizations device hierarchy in the attempt of login to applications. This is one of the unique feature comes with Universal BioSys. This tree structure is maintained in a XML document and also a record is saved to the database (this data is useful in validating the hierarchy) with the parent device and child device relationship. XML format is used to maintain the tree structure and it can be seen through the Administrative console. XML parse [section 2.8] is used to parse this XML document and enter the data to the database. For that SAX parser is used which comes with the XmlTextReader class in Microsoft .NET framework.

When a device is allocated to a user for submitting his/her biometric data that device is temporally blocked by the system. These blocked device details are stored in the BioSysManager session and apart from that current active logins are also stored in a session variable. Let's consider the verification scenario, where a user wants to prove his application that he is, who is he claims to be. The user first types his user name and requests a device to submit biometric data. BioSysManger first checks whether user has enrolled to the application with the selected device type. If the user has been enrolled then service selects the available devices in the device type, then search in the current active



logins to find the previously authenticated device for the user. If any device exists then find a child device of that device from the available un-blocked device list. Otherwise select a device from the device list which does not violate the device tree structure. To select the optimum device, location of the host is also sent to the BioSysManager. So the service selects the closest device with respect to given host's location. In the identification mode BioSysManager select the closest device with respect to the host and let the user to submit the biometric identification. In identification more than one user will get matched and all the matched users are passed to the client side (best matches users are given at the top of the list).

BioSysManager maintains host and device locations with the details of rooms, floors and the x, y coordinates of the floor. According to the all these parameters BioSysManager select the best device.

When user submit his/her biometric record, all the image processing related actions are done at the server and then it informs the client whether user is authenticated to the application or not. BioSysManager maintain information about user status (when did he/she logged in, where did he login what is the application being used, etc.) and those records are used for administrative purposes such as to track the current user location within the organization and forcefully logout a particular user from the system.

BioSysManager provides all the required features for the Universal BioSys to be a marketable product and clients to acquire distinct services.

#### **4.1.2 Administrative Console**

Administrative Console is the major client connected to the BioSysManager which is a window based application (a thin client that only performs data validation) and it includes a user guide as well. All the management, monitoring and administration are done through this component. Applications, device types, devices and hosts details are



maintained by the console. Administrative Console can be used separately from the BioSysManager either on the same machine as the server or as a remote application installed in different host.

Console operates on two modes; the default mode is the read-only mode where the administrator can only monitor activities of the system. If he needs more configuration privileges he has to change to the privilege mode by authenticating through a biometric device (password is not supported at this stage).

To gather location information of the devices and hosts organizational floor layouts are provided. Organizations that span multiple floors can also assign policies as different layers. User enrolling, updating, deleting and banning are performed through the Administrative Console. In the enrolment user can be enrolled to an application with one or more device types include giving distinct usernames. Current user states and also past user logins can be viewed by the administrator through the console. Administrator has the privilege to logout users form the service that they are already logged in.

User location tracking is another important feature supported by the console which is really essential in large an organization that spans several building or floors of the same building. This feature allows the administrator to track the user location and it is indicated through a map.

As most other solutions Universal BioSys keeps extensive of log of activities. Two types of logs are supported by the system: event log (keep track of events) and system logs (keep track of system status). These logs and can be analysed through the console.

Administrative console is developed using C# language and it can run on any Windows machine that contains the .Net framework. Administrative Console is flexible enough and it can be changes according to the requirements of the administrator. Administrative



Consol for Linux environment can be developed in Java with using the Java packages provided by Universal BioSys.

#### **4.1.3 Application Client**

This is a component that communicates with the BioSysManager which enable its application to verify or identify its users. Application client is provided as a COM component, where the application developer has to use as a reference during the development stage. Through this the developers can call the verification and identification remote methods in the BioSysManager.

Application clients can be easily built from any SOAP supported languages or application client functionalities can be implemented inside the application by the developer with importing the BioSysManager web service. Because of these features application client functionalities can be implemented independent from the language and the platform.

Component obtains the application ID and gets the device types available for the application. Then user can select the device type (including password) which he wants to submit the biometric identification. If the user is entitled to user the device type and unoccupied devices are available from the particular device type then BioSysManager blocks the best selected device and send the device details to the client side. Application client has the functionality of showing the selected device location in a floor layout with the device name which makes easy for user to locate to the correct device.

In the verification mode application client informs the application whether user is authorized to use the application or not with the username and in the identification mode it informs the application the list of username who are authorized by the BioSysManager server. The best authorized username is at the top of the list.

Application client is a layer between the applications registered in the Universal BioSys and the BioSysManager which makes the developers' tasks are easy in authorizing users to the developed application.

#### **4.1.4 Desktop Client**

This is an application which should be installed on every host that is supposed to work with Universal BioSys. When it runs for the first time it will contact the BioSys server and register it self with the server. With the registration, machine obtains an UUID from the server. All the application for the machine can be viewed from this client application. When user wants to log out from the application he can use the feature provided by this module. This is a background program which starts up when machine boots up. Present client module is done using C# but with the java classes provided as a package will enable the same functionalities.

Desktop Client is installed on Microsoft Windows machines with a customised Installation wizard. Desktop Client will reside at the taskbar in Microsoft Windows and for other operating systems java based client can be developed using the special package provided.

## **4.2 Cost effective Face Recognition**

As a part of our project we developed a fully pledged Face Recognition system. It serves two purposes in our project.

1. To demonstrate the functionality of Universal BioSys. An actual Biometric device will make the project demo more lively and real.
2. As the Face recognizer is cheaper and fairly accurate, it can be used with Universal BioSys in an actual environment. Therefore someone who deploys



universal BioSys does not need to wait for a BioAPI compliant device. Hence Universal BioSys is more complete with the Face Recognizer.

3. The Face Recognizer can be used as product, independent from Universal BioSys. That is, it can be used as the face recognition device in another solution.

This section concentrates on the facial recognition in general, the method we adopted and its pros and cons.

#### 4.2.1 Introduction

Face recognition uses the visible physical structure of an individual's face for recognition purposes. A typical face recognition system is explained below to illustrate the concept behind the rest of the chapter.

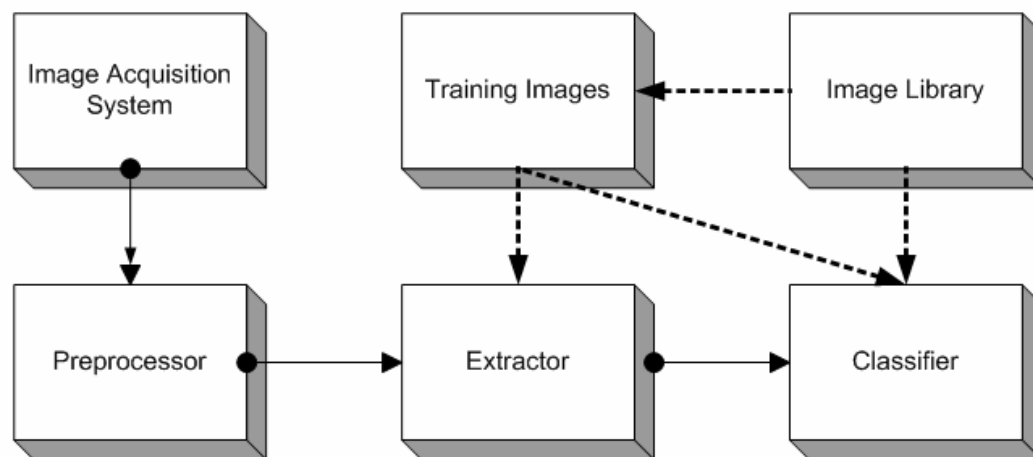


Figure 4.3 – Components of a typical Image processing system

Image acquisition system	This is the entry point of the face recognition process. The user presents an image to the system. An image capturing device is used to capture the image into digital format, like the web cam in our case.
Pre-processing module	Face images are enhanced to improve the recognition performance of the system. For example, noise (unwanted data present in the image, due to environment or due to defects in the image acquisition system) is removed.

Feature extraction module	This module is responsible for composing a feature vector that is well enough to represent the face image.
Classification module	Extracted features of the face image are compared with the ones stored in a face library (or face database). After doing this comparison, face image is classified as either known or unknown.
Training set	Training sets are used during the "learning phase" of the face recognition process. The feature extraction and the classification modules adjust their parameters in order to achieve optimum recognition performance by making use of training sets.
Face library or face database	Face images are added to a library with their feature vectors for later comparisons.

#### 4.2.2 Key Features of the Facial Recognition System

It's cost effective. Images are captured through a low quality web cam costing 50 US\$.

The web cam offers a resolution of  $320 \times 240$  (without interpolation). Many popular face recognizing algorithms will prove inaccurate under these conditions. Hence the Eigen face approach, as explained later in the chapter is used.

The Facial Recognition system supports identification in addition to verification.

#### 4.2.3 Image Acquisition

DirectX is used to capture images through the web cam. The GUI of the image capturing software was developed using MFC. It was not easy to develop a rich GUI using Visual C++. But finally we managed to do so. DirectX proved to be very powerful and useful than we imagined. After studying several samples shipped with DirectX, we managed to implement the image capturing functionality reusing DirectX libraries and sample code.



#### 4.2.4 Pre Processing

The captured image was subjected to several image processing operations.

Noise filtering was applied to remove any undue noise. Median Filtering was preferred over other filters as it does not blur the image. In object recognition, it's very important that images are sharp as much as possible.

The noise filtered image was sent through a background removal operation (i.e. remove the part of the image that does not include the object of interest, that is the face) we developed.

#### 4.2.5 Eigen Face, Face Recognition

This was the critical and complex area of the face recognizing system. We must thank Mr. Chathura for suggesting Eigen faces for face recognition, when we were struggling with geometric feature based matching. To better understand the Eigen face concept, an insight into face recognition algorithms is given in the next few sub sections.

##### 4.2.5.1 Face Recognition Algorithms

Face recognition is a very complex activity in human brains without any solid explanation. Recognizing faces is different from finding faces. Face recognition is about describing who is the person, where as face detection is about locating a face in an input image or in a video sequence.

Many algorithms work as a two step process. First the image is projected in to a subspace (stored images make up the universal set spanning many sub spaces). Then a classification algorithm is used to identify the projected image in the context of the subspace.

Face recognition algorithms fall in to two categories. PCA (Principal Component Analysis) method is based on the information theory concepts. In this approach, the only most relevant information that best describes a face is derived from the entire face image. The second method is based on extracting feature vectors from the basic parts of a face such as eyes, nose, mouth, and chin. In this method, with the help of deformable templates and extensive mathematics, key information from the basic parts of a face is gathered and then converted into a feature vector.

#### **4.2.5.2 Dimensionality Reduction**

An image space has dimensions equal to the number of pixels in the image. Each dimension has values in the range of the pixels values. Thus, for example a grey scale image of size (  $M \times N$  ) has the dimension of (  $M \times N$  ). In the case of grayscale images, in each dimension the image could have a value in between 0 and 255 (0x00 to 0xFF).

Computers need a to store a sufficient number of different views associated with each other object, for accurate recognition, and this requires a large memory space. Therefore it is required to reduce the storage per each stored item. That is dimensionality reduction.

#### **4.2.5.3 PCA**

Central idea of PCA is to reduce the dimensionality of a data set, having a large number of interrelated variables, but retaining as much as possible of the variation present in the data set. Dimension reduction is achieved by transforming to a new set of variables (principal components, PCs), which are uncorrelated and ordered (so that the first few retain most of the variation).

Principal component analysis, when applied to facial recognition is based on information theory concepts, and seek a computational model that best describes a face, by extracting the most relevant information contained in that face.

#### **4.2.5.4 History**

Kirby and Sirovich were among the first to apply PCA to face images. Turk and Pentland popularized the use of PCA for face recognition

#### **4.2.5.5 Eigen Faces Algorithm**

Eigen face approach is a variation of PCA.

The scheme decomposes face images into a set of characteristic feature images called eigenfaces, which may be thought of as the principal components of the initial training set of face images. Recognition is performed by projecting a new image onto the subspace spanned by the eigenfaces and then classifying the face by comparing its position in the face space with the positions of known individuals.

Eigen face method is based on the correlation, thereby the variation between a set of images. The face images when converted into vectors, will group at a certain location in the image space due to the similar structure, each having eye, nose and mouth in common and their relative position correlated. The Eigenface method seeks to find a lower dimensional space for the representation of the face images by eliminating the variance due to non-face images by focusing only on the variation between the face images.

Complex mathematics is involved in calculating Eigen faces. Once the Eigen faces are available, a template for each library member needs to be calculated. The theory behind template calculation is simple. Each library image is a linear combination of the principal component (i.e. Eigen faces). Mathematically speaking each image  $I_i$  is a linear

combination of the set of Eigen faces (K no of Eigen faces)  $\mathbf{n} = [\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3, \dots, \mathbf{n}_K]$ , and  $[w_1, w_2, \dots, w_K]$  are the weights in the particular linear combination.

$$I_i = w_1 \mathbf{n}_1 + w_2 \mathbf{n}_2 + \dots + w_K \mathbf{n}_K$$

Wight vector associated with each library image uniquely identifies each library image.

Therefore each face is represented by two entries in the face library: One entry corresponds to the face image itself and the other corresponds to the weight vector associated with each face image.

Eigenfaces approach seems to be an adequate method to be used in face recognition due to its simplicity, speed and learning capability.

#### 4.2.5.5.1 Eigen Face Calculation

An image can be thought as a vector in the image space through the concatenation of each column of the image one after the other.

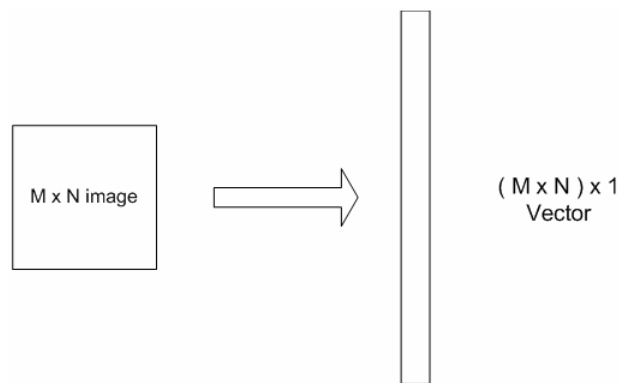


Figure 4.4 – Vector representation of an image

Let's denote an image by  $I_i$  which is  $(M \times N) \times 1$  vector.

The set of images (K no of images)  $X = [I_1, I_2, I_3, \dots, I_K]$

In the Eigen face method, the features of the studied images are obtained by looking for the maximum deviation of each image from the mean image.

Therefore the mean image  $\Psi$  is calculated as  $\Psi = \frac{1}{K} \sum_{i=1}^{i=K} I_i$

Then the deviation from the mean is calculated as  $\Phi_i$ , which is the deviation of image  $I_i$  from the mean image.

$$\Phi_i = I_i - \Psi$$

Thus the set of mean subtracted images are calculated

$$A = [\Phi_1, \Phi_2, \Phi_3, \dots, \Phi_K]$$

This variance is obtained by getting the Eigen vectors of the covariance matrix of the calculated mean subtracted images.

$$\text{Covariance matrix} = AA^T$$

But the resulting covariance matrix is huge for a considerable no of images, as explained below.

The dimension of  $A = (M \times N) \times K$

And thereby  $A^T = K \times (M \times N)$

Hence  $AA^T$  will have dimension of  $(M \times N) \times (M \times N)$

Our web cam provides a resolution of  $320 \times 240$

Thus in our case  $M=320$  and  $N=240$

Therefore dimension of  $AA^T = 76800 \times 76800$

It will be computationally intensive to find the Eigen values of this huge matrix.

But fortunately through the Eigen values of  $A^T A$ , Eigen values of  $AA^T$  can be calculated, without loss of accuracy. There's an elegant proof to support it [9]. Since  $A^T A$  is of dimension  $K \times K$ , and typically the image base will be very much less than 76800, a huge amount of computation can be saved.

The eigenvectors are produced such that the first one captures the most variance in the data, the second captures most of the variance once the first has been removed, and so on.

Once we calculate the Eigen vectors of  $AA^T$ , each of these vectors is an Eigen face.

The word Eigen face comes from the fact that they are generated from Eigen vectors and have a face like appearance.

Let  $\mathbf{n}$  be the set of Eigen faces

$$\mathbf{n} = [\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3, \dots, \mathbf{n}_K]$$

Out of these, Eigen faces that have the largest Eigen values account for the most variance within the set of library face images. Therefore only the best Eigen faces needs to be used, without a significant loss of accuracy. The Eigen values are ordered in descending order. Thereafter through experimental procedures, one can determine the no of Eigen values (vectors), to be considered without significant loss of accuracy in the context of that application. The best E Eigen faces span an E-dimensional subspace, which we call the "face space" of all possible images.

This set of best vectors (i.e. Best Eigen faces) will be saved to the hard disk, for encoding of the library images, as explained in the next sub section.

Please note that as new faces are experienced, the eigenfaces needs to be recalculated

#### 4.2.5.5.2 Encoding of Faces

The concept behind encoding of library images is that each individual face can be represented exactly in terms of a linear combination of the Eigen faces.

Therefore the next step is to project each image (mean subtracted image) into the Eigen vector space.

The projection is identified by a set of coefficients referred as weights and this set of coefficients is the encoding.

Let Weights of the  $I_i$  (actually the projection of  $\Phi_i$ ) be

$$W_i = \mathbf{n}^T \Phi_i$$

Therefore the weights matrix  $W$ , which is the matrix of weight vectors is given by

$$W = \mathbf{n}^T A$$

#### 4.2.5.5.3 Recognition and Learning Phase

After the construction of weight vectors of face library members, now the system is ready to perform the recognition process. In the Eigen faces approach, recognition is very obvious and simple. The mean image is subtracted from the test image and projected into the Eigen face vector space.. After obtaining the weight vector, it is compared with the weight vector of every face library member within a user defined "threshold". The method of comparison we have used is the Euclidian distance. For one stored weight vector, the difference between each weight and the corresponding test user weight is calculated, squared. These squared values are summed to obtain the Euclidian distance

### 4.2.6 Services of the Facial Recognition System

#### 4.2.6.1 Enrolment

Eigen faces approach is quite different to geometric feature based methods. In those methods, at each enrolment, geometric features of the new image are examined and a template is saved.

Thereby each image is independent. But in the Eigen face method, at each enrolment a new set of Eigen faces needs to be built, as explained above. The new Eigen faces are

built from the existing images with the inclusion of the new image. Then the weight vectors of each and every image will be recalculated. This interdependence between images posed a major problem to us.

The biggest issue was BioAPI expects a single template as the result of enrolment. But in the Eigen faces approach the result of an enrolment is mainly the set of Eigen faces and the weight vectors. Since only one template can be returned, there was no other option other than to save the Eigen faces locally at the BSP. But to calculate Eigen faces the system frequently requires all the existing images. The BioAPI supplies a processed image to the enrolment function and there is no room to pass existing images. Hence it was decided that all the images will be saved at the BSP as well.

Theoretically the template that is saved for each user in the database is a set of double coefficients. (A vector of doubles) and per each enrolment, weight vectors of each library member needs to be recalculated and returned. But, BioAPI expects a set of bytes (a pointer to bytes). Because each image pixel is a Byte (if it's grayscale) or an integer multiple of Bytes (if its colour). Thereby if we submit a set of doubles (where the number of doubles may vary) it might be difficult to reconstruct the doubles at the receiving end. Therefore we removed the coefficient determination stage from the enrolment function. Hence the final decision was to calculate the Eigen faces from the input processed image, save locally and resend the input back as the enrolment result.

#### **4.2.6.2 Identification**

The input to the identification function is a set of stored images plus the image of the test user. The system tries to identify the best match for the test image from the stored images. The logic of identification is very clear with traditional geometric based approaches. Simply calculate the template of the test image and compare with stored templates. And



to create templates of test users, the server does not need existing images. With the Eigen face approach its different, the template is the weight vector, and to calculate it the set of Eigen faces is required. Thereby the locally saved Eigen faces are read from the hard disk.

Weight vectors of each of these images are calculated using the read Eigen faces.

Then the test image is classified using the Euclidean distance between its weight vector and the weight vectors of the stored images. (The stored image with the least Euclidian distance is identified). Verification is very much similar, where only one stored image is inputted.

#### **4.2.7 Limitations**

The Facial recognition system uses a low quality web cam, for image capturing. Therefore the Eigen Face approach is the best suited, and thereby used in our project. Eigen Face approach has several drawbacks. It is susceptible to changes in faces, which is typical with age. Accuracy of any face recognition algorithm fails with changes in the stored images, but Eigen face method is affected severely. At each enrolment, all the stored templates need to be recalculated, which is another drawback, but verification and identification works on simple and small templates, enjoying a significant speed factor. As enrolment is less frequent, it is not that much of an issue.

The BioSys team hopes to improve the facial recognition system as an independent product, a low cost alternative to facial recognition. Therefore more pre-processing operations can be included to improve the accuracy of the recognition.

To conclude, the BioSys team is proud to develop a fully functional Facial Recognition system with a comparatively low cost of ownership, as a part of the project.

# Discussion

## Results

Being a proof of concept, the Universal BioSys does not provide measurable results.

We are proud to say Universal BioSys is a working solution. Though, not directly marketable, functionality wise it is complete

Several measurable results can be shown for the Facial Recognition system.

## Future Directions

Our goal was to demonstrate that it is possible to devise a solution in which any type of biometric device can be seamlessly integrated and centrally managed in a networked environment and accessed through a myriad of applications without the knowledge of the specifics of each and every device. But our solution is far beyond from a total security solution. Through this six months (with only two months of actual development) effort we have proved the concept, but a lot needs to be done to make it readily marketable.

A charismatic and a vital characteristic of a total security solution would be in-depth security with in the solution itself. That is, the security solution itself should not be the vulnerability. Hence the solution should be immune to any type of attack ranging from simple ping floods to strategic virus attacks (caused by buffer overflow attacks). Security should be built from ground up. It is impossible to implement solid security by wrapping up an existing system. It takes special effort and dedicated time to incorporate security from ground. It is difficult, if not impossible to ascertain this level of security with in six months considering the amount of functionality given by BioSys. Hence Universal BioSys may need to be revised substantially to manifest defence in-depth.



BCB Generator (BioAPI Compliant BSP Generator) is another prospective area of development. At the start of the project we understood the immensity of the project scope and decided to limit the BCB Generator just to the specification. Thereby we have identified its potential and the functionalities, and its implementation is a future work. BCB Generator generates fully BioAPI-compliant BSPs for non-standard biometric devices given specific device information (device interfacing source code). It enables vendors of biometric hardware devices and biometric algorithms to easily create fully BioAPI-compliant biometric service providers (BSPs). We have identified several functionalities that are common for any type of BSP. These will be developed as core modules of the target BSP. Then upon the algorithms and device specific information given our BSP builder will generate custom modules for each target BSP. Then the custom modules can be allowed to change to some extent, as there can be very propriety but necessary features for a certain BSP. Finally both the core modules and the custom modules will be shipped as the BSP. The BCB Builder supports all of the mandatory and optional features of BSPs described in the BioAPI 1.1 standard. The availability of BCB Builder will result in wider acceptance and conformity of BioAPI adding more value to Universal BioSys as well.

Universal BioSys can be extended to incorporate several features offered by most network related tools. One such feature would be extensive report generation. All the exceptional events (e.g. attempts to bypass the device hierarchy) can be generated as a report and e-mailed to the system administrator. Not only e-mail but also other communication means like SMS, or system generated alarms can be supported. More value can be added by combining with a popular domain management system like Microsoft Active Directory. Other than the security mechanisms to protect the Universal BioSys, which we have mentioned in earlier paragraphs, fault tolerance features needs to be implemented to ensure the availability of services.

Universal BioSys can be used by any application that requires biometric services. We can develop several such applications and ship with Universal BioSys with an additional charge for each such application. One such application would be a time and attendance system. An organization that uses Universal BioSys would prefer a time attendance system in the same package, than to develop separate software through another software vendor. Working along the same concept, another area of extensibility would be the integration with a ticketing server like Kerberos. Therefore users who identify or verify through Universal BioSys can obtain tickets to use certain applications for the time period stated in the ticket.

Universal BioSys is a functionally complete solution. In this section the BioSys team has identified many areas of improvement to improve the value of Universal BioSys, as future directions.

## Conclusion

With the rapid advancement in technology, automation has changed our lives dramatically. It is really regretting that in this modern world where refrigerators self-order when the stocks run out, people still carry tokens or have to remember long passwords.

There are dozen of technologies, well advanced but affordable, where the human himself can be the access token. This authentication suite is generally referred as biometrics. Despite the hype biometrics has not gained wide spread usage. The reason is the difficulty of integration with in a networked environment. Every biometric vendor has his own propriety implementation. Each of these products comes with it its own SDK, for application developers. Hence if an application requires using two types of biometric devices, maybe to accommodate different levels of security, the application developer needs to devise his application to support both SDKs. Furthermore if multiple users at different locations require authentications services the problem is devastating. The obvious approach then would be to install a device at each and every location, which is the standard approach prevalent today. Though technically simple, it is a huge cost to incur. Thus it is essential to share several devices through a networked environment, and this is too much a task with existing solutions.

Universal BioSys is an elegant solution that addresses this issue. Its core is the BioAPI a well formulated and accepted standard. Once a BioAPI compliant biometric device is installed Universal BioSys can read its configurations and understand its capabilities. Thereafter enrolment, verification and identification services of this device are exposed through a web service which we refer as the BioSys Service. To make it professional and powerful an ActiveX component is provided to the application developers to access the published web services. Off the site (not off line) Administrative console is provided to



set the parameters (i.e. configurations, policies) and monitor the activities of the whole solution. Finally to make it more complete, a fully pledged facial recognition device is shipped. This low cost device is can be used seamlessly with Universal BioSys server and demonstrates what BioAPI compliance means.

### **Advantages and Disadvantages of Universal BioSys**

Many operational and economical advantages can be gained by adopting our solution.

#### **Economic advantages**

With current normal practices, each application that requires biometric services will require special development effort for the biometric integration (requires high skilled developers who are good in C/C++). Universal BioSys can save development costs substantially by providing a simple to use interface, without any fuss of memory management and pointer handling.

Furthermore due to the difficulty of integrating various biometric devices, organizations tend to use one type of device through out the network regardless of the cost.

And if the biometric device is changed a high refactoring cost will be incurred.

Therefore Universal BioSys will definitely bring down the TCO with its simple to use and implementation independent interface.

#### **Operational Advantages**

Three types of user's benefit from Universal BioSys

*The Administrator:* Is relived of many mundane tasks. All the biometric devices spread through out the network can be managed, through a central location. Seamless integration with familiar administrative tools and procedures make it easy to administer with his all



other day to day duties. Universal BioSys is deployed through standalone setups. Hence it is a packaged solution that installs into the existing environment reducing the installation time and effort of the administrator.

*The Application Developer:* So far the application developer was required to have an understanding of the biometric device they use. Even if it's a MIS application, the developer had to be burdened with these unrelated issues. Even with BioAPI it is not that easy with complex data structures, pointer handling and wearisome memory management, as we experienced. With our solution, it's very easy for the application developer. In his application he just needs to call web services (or use the ActiveX control, which simplifies the matters more) and he is relieved of all the biometric related features. Therefore he can concentrate on the application he is supposed to develop resulting in better productivity. So it's obvious that any application developer is going to welcome our solution with much warmth.

*The user:* Since all the above features of our solution, the user will be assured consistent and much easier identification /verification. Even if one device is unavailable, requires no waiting they can use another device in the network. And the prime factor is they can forget their password!

Therefore its obvious Universal BioSys is not just any hype, but an immense value proposition. The mere fact that all its services are published as web services makes Universal BioSys scalable to the internet. Hence an organization deploying our solution can let a user log in from a registered device anywhere in the internet.

Universal BioSys is not a total security solution. The fact needs to be stressed as an observer may expect several standard features in a security solution, but not currently available in our solution. The reason is as explained several times; Universal BioSys is a proof of concept. A lot of revisions need to take place to make this proof of concept marketable. In depth security is a must for any security solution. Universal BioSys does



not meet this standard level of security. It's possible to devise attacks on the server and be successful. The scope was too much, to incorporate hardened security, which the BioSys team understood from the beginning it self.

BCB Builder is another area we need to pay attention in the future. At the beginning we decided not to implement it as we will not have enough time to complete it. Therefore it was limited to the specification as planned.

In addition to above leftovers, there were several requirements initially set out by us, we were unable to meet. They are not core features, but functionalities we planned to have in the admin console and the ActiveX component. For example, several real time monitoring features were left unimplemented.

Though Universal BioSys is not ready for the market all the hard ground work has been done and the concrete base is flexible and scalable to be a competitive solution in the biometric security arena. We hope that our effort will lead to an era of biometrics, a world without passwords and bulky magnetic and crypto cards.



## References

- [1] Universal BioSys web Site  
<http://www.biosys.net.tc>
- [2] The BioAPI Consortium, "BioAPI Specification Version 1.1", 16<sup>th</sup> March 2004.  
<http://www.bioapi.org>
- [3] The Independent Security Server, developed by Info Data, Inc.  
<http://www.infodatany.com/independentsecurityserver.htm>
- [4] The WhoIsIt biometric server for E-commerce  
[http://www.qvbiometrics.com/E\\_Metrics\\_server.htm](http://www.qvbiometrics.com/E_Metrics_server.htm)
- [5] De Silva S. M. R. P , Weerasinghe P. W. H. D, Bandara H. M.N. D, "Universal BioSys - A literature review"  
<http://www.cse.mrt.ac.lk/~ravids/literal.html>
- [6] Accredited Standards Committee  
<http://www.x9.org>
- [7] Simon Robinson, K. Sott Allen, Ollie Cornes, Jay Glynn, Zach Greenvoss, Burton Harvey, Christian Nagel, Marogan Skinner, Karli Watson, "Professional C# 2<sup>nd</sup> Edition", Wrox Publications, ISBN 81-7366-452-8
- [8] Cay S. Horstmann and Gay Cornell, "Core Java 2 Advanced Features", Person Education, ISBN 81-7808-846-0
- [9] Ilker Atalay, "Face Recognition using Eigenfaces", January, 1996  
<http://www.ilkeratalay.com/facerecognition.php>



# Appendix A

## Development Process

The Universal BioSys team approached the project in a quite a different manner compared to most other projects. Many of the teams were on to coding within one or two months of the entire project life. The team decided it was not the best method in our context. The big challenge of Universal BioSys is to identify the functionalities of this proof of concept and to come up with an elegant design. Therefore major effort should be put on analysis and design.

Accordingly the first five months of the project was entirely dedicated for analysis and design. Not a single line of program code was written during that time. Many long hours were spent on discussing and debating about the requirements and design. There was a powerful and valid reason behind this particular approach followed by the Universal BioSys team. Universal BioSys is a proof of concept, and the concept itself is novel. Therefore greater effort should be spent on defining the concept, and designing the best way of proving the concept. Otherwise implementing an impractical, unusable concept or a bad design is a loss of time and effort. Normally this is not the case, as many projects are working on well defined project ideas (mainly RFCs, specific business or scientific requirement).

The BioSys team worked according to this plan, and now at the completion of the project, think it was the best way, to be followed. The team was able to identify many functionalities and their pros and cons, and designed for them. Therefore the concept is more or less complete. Furthermore during the analysis and design stage, the team documented all their work in a number of artefacts. Therefore the team is proud to say,



that they have a well documented analysis and design. All of these artefacts are downloadable from the BioSys website [1].

Then as the team had a solid design, with in two months of development nearly all the functionalities were implemented. The only unimplemented features were non-core features, mainly non-functional requirements like the rock solid security within the solution.

The BioSys team feels the project was entirely a success. It was through sheer dedication and the harmony between members. The team members feels the final year project was one of the milestones in their lives, as it learnt them team spirit, patience and perseverance in addition to technical facts.

## Appendix B

### Detailed Design

The design approach of the Universal BioSys was followed based on the UML (Unified Modelling Language) concepts. Use Case diagram were first drawn to identify the use cases of the system and main functionalities involved in .

#### Use Case Diagram

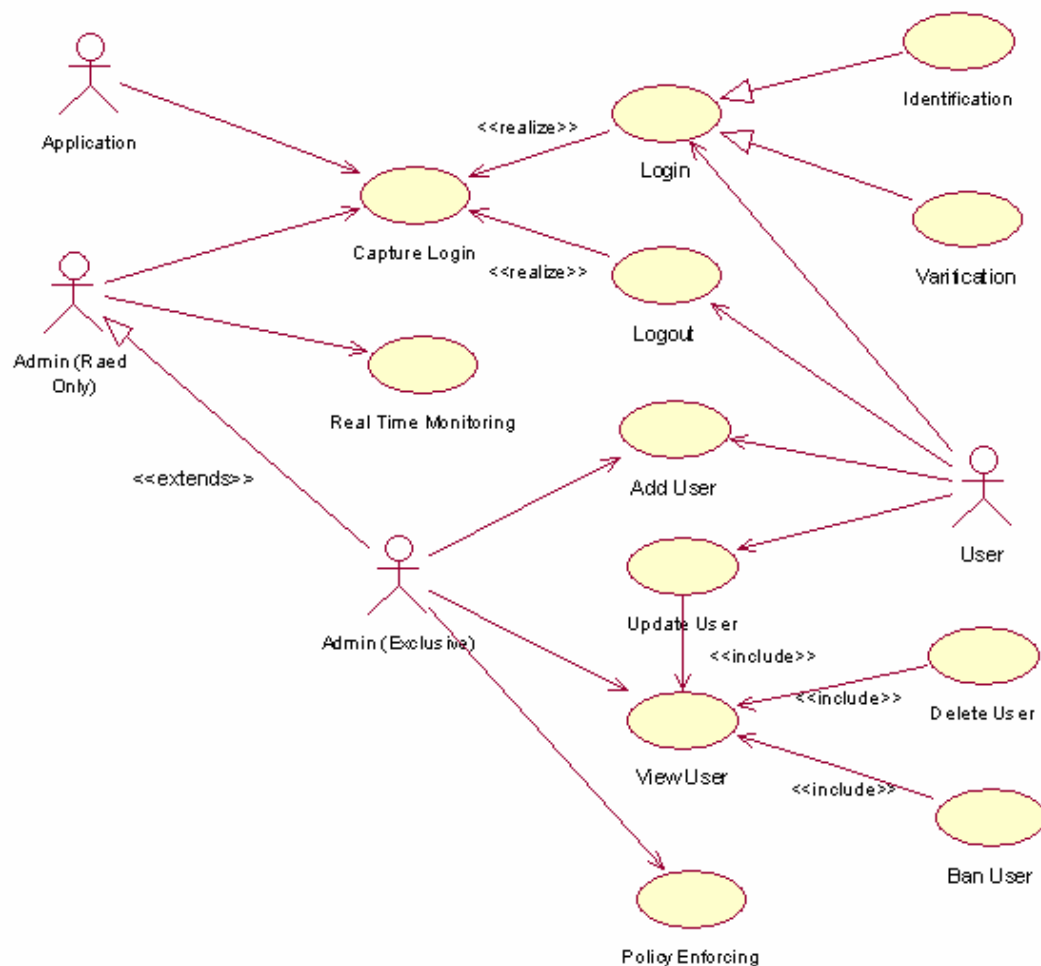


Figure B.1 – Use Case diagram

## Activity Diagrams

Then Activity diagrams were created based on the above Use Cases.

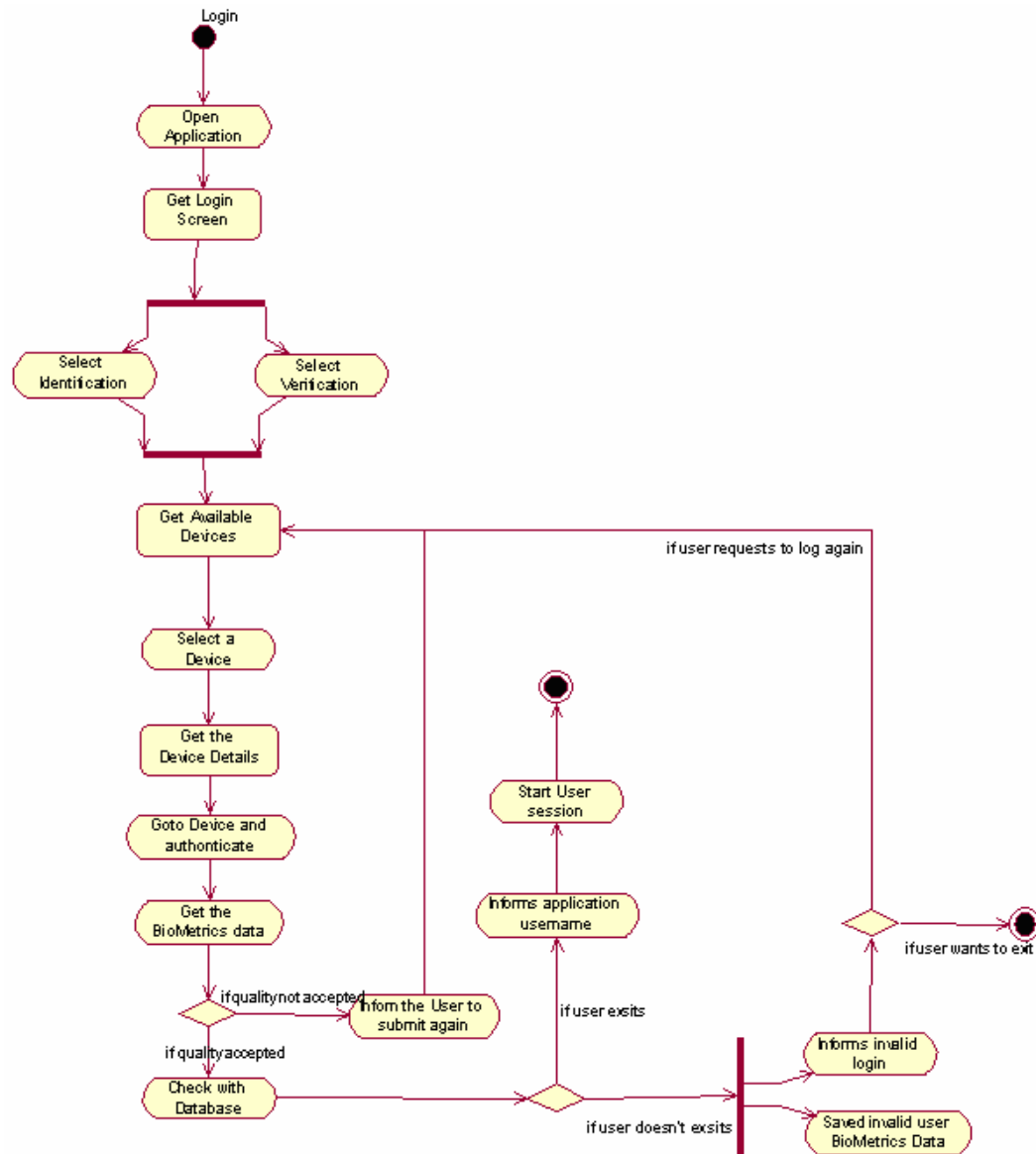


Figure B.2 – Activity Diagram: User login

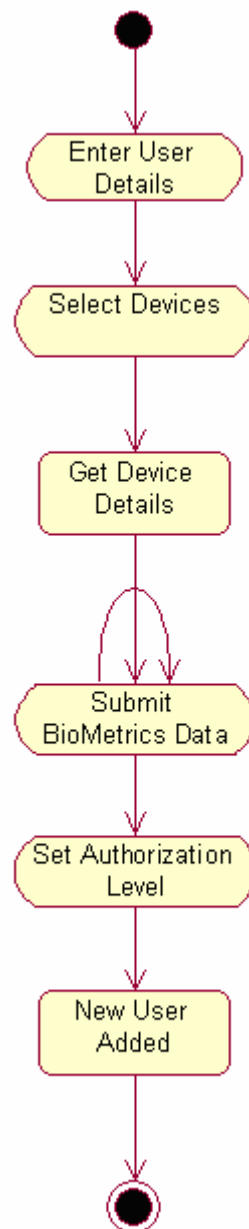


Figure B.3 – Activity Diagram: Add user

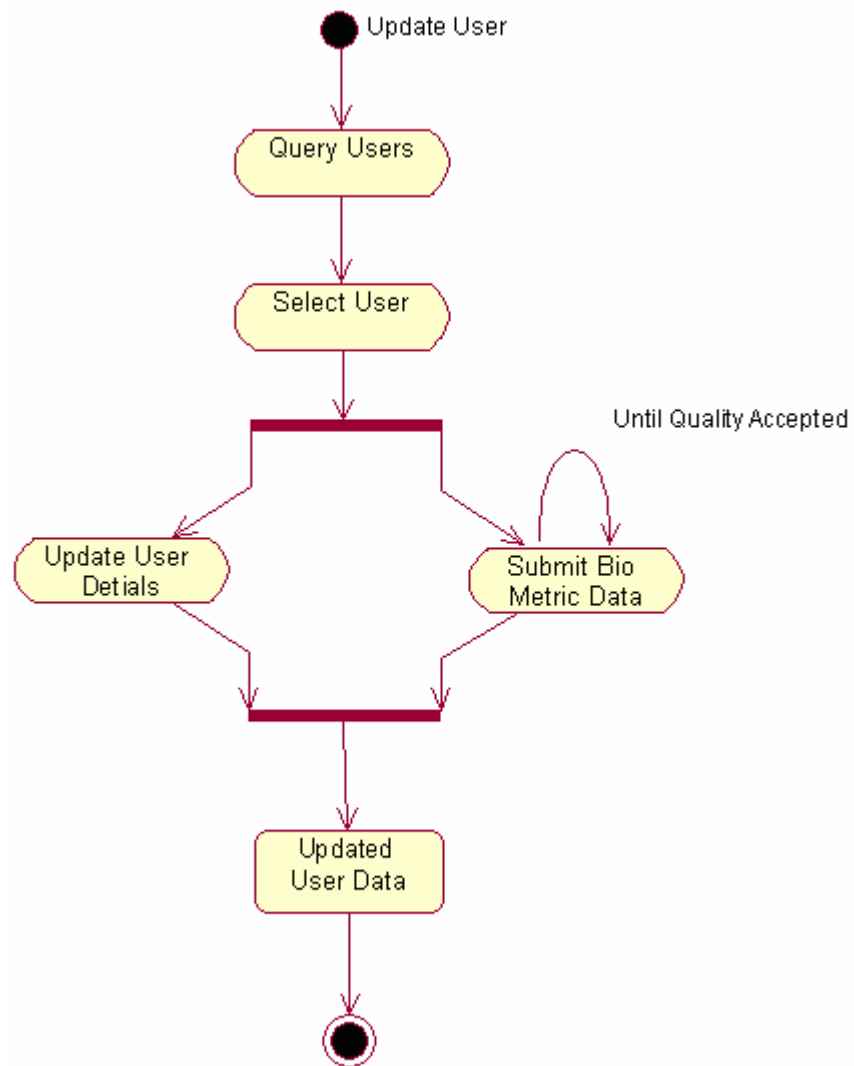


Figure B.4 – Activity Diagram: Update user

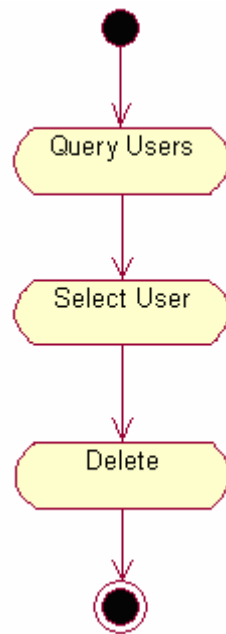


Figure B.5 – Activity Diagram: Delete user

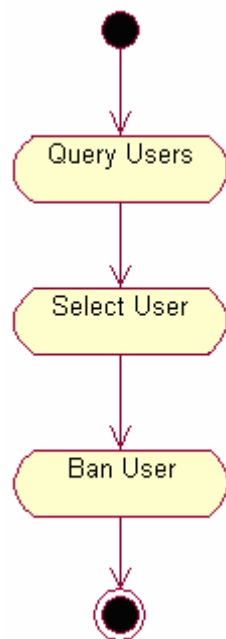


Figure B.6 – Activity Diagram: Ban user



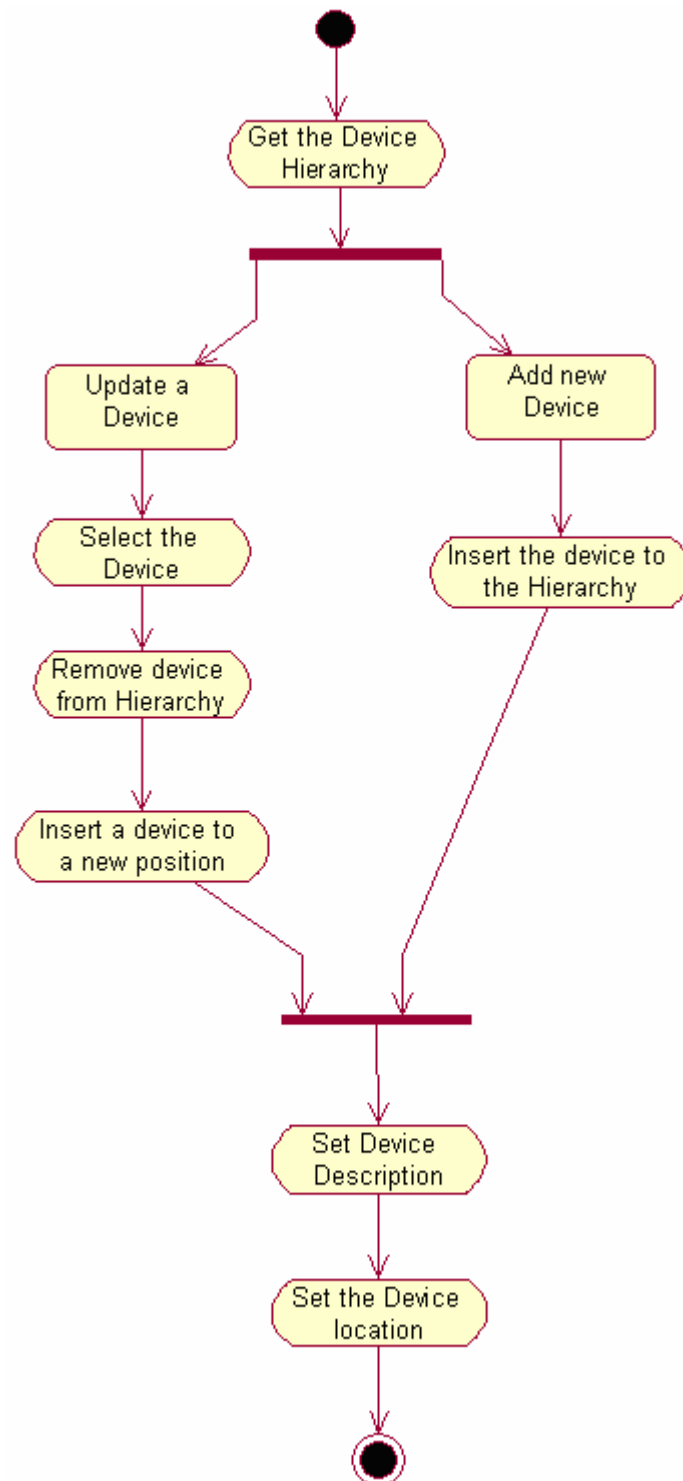


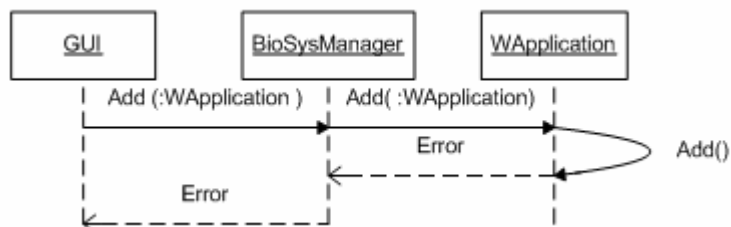
Figure B.7 – Activity Diagram: Policy enforcement



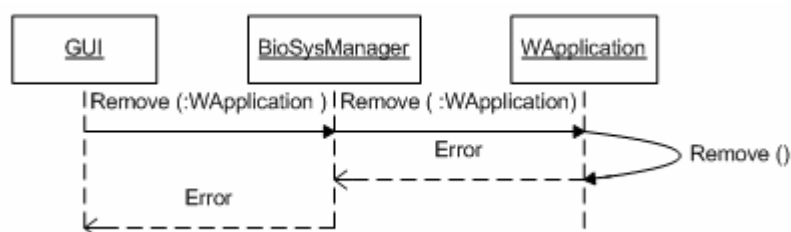
## Sequence Diagrams

Sequence diagrams were designed as the final step of the UML approach.

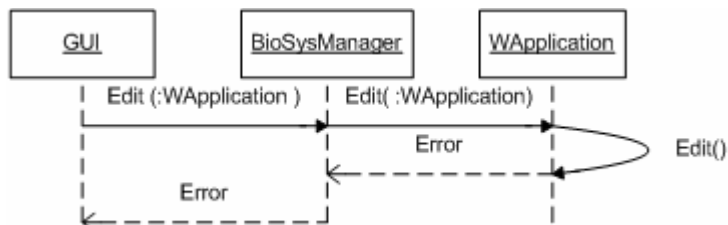
### Add application



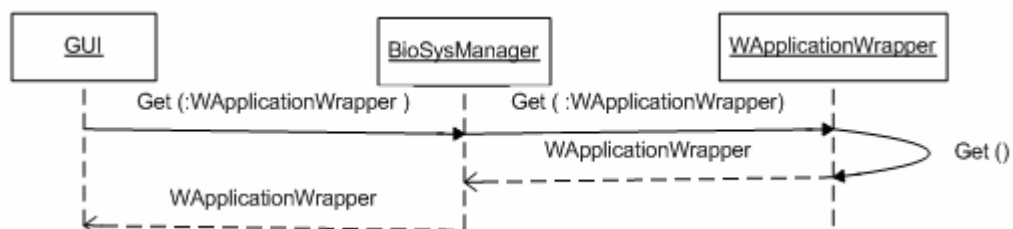
### Remove application

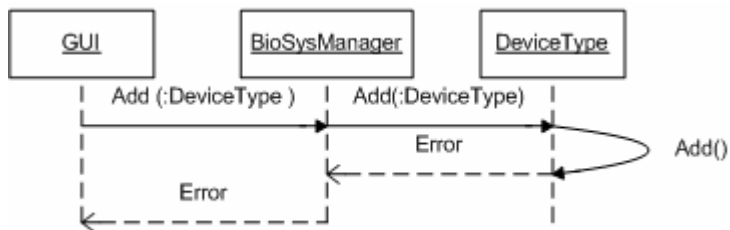
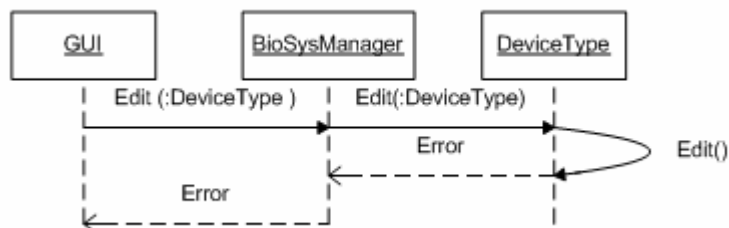
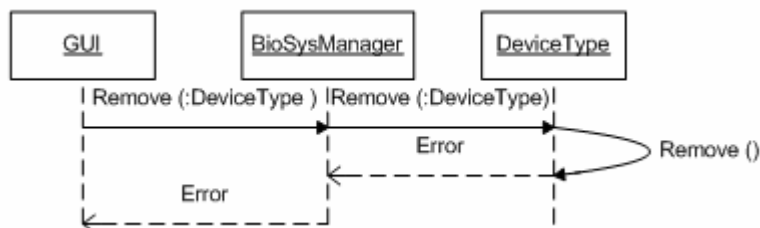
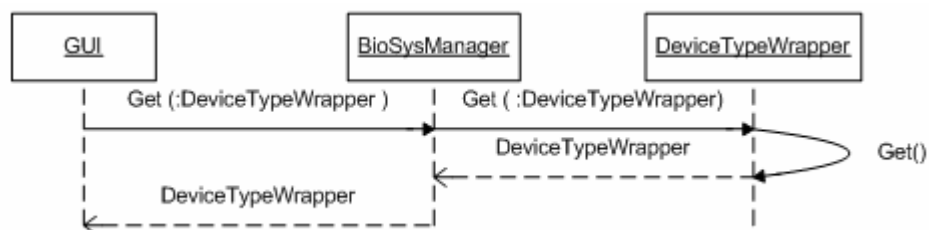


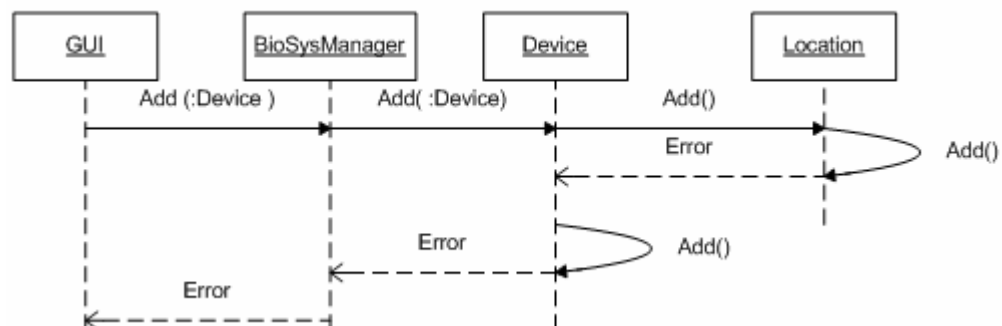
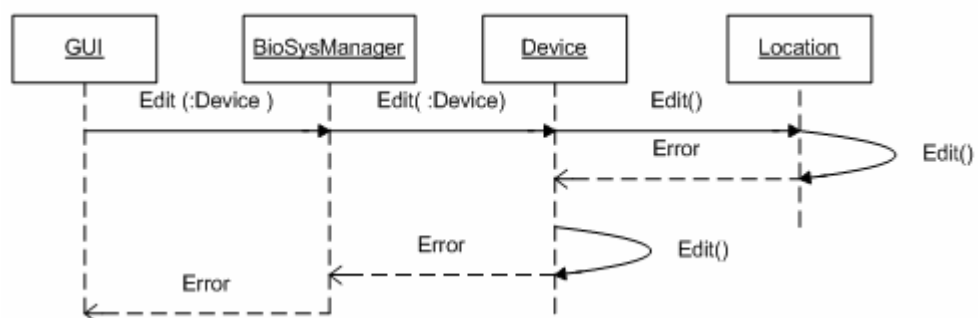
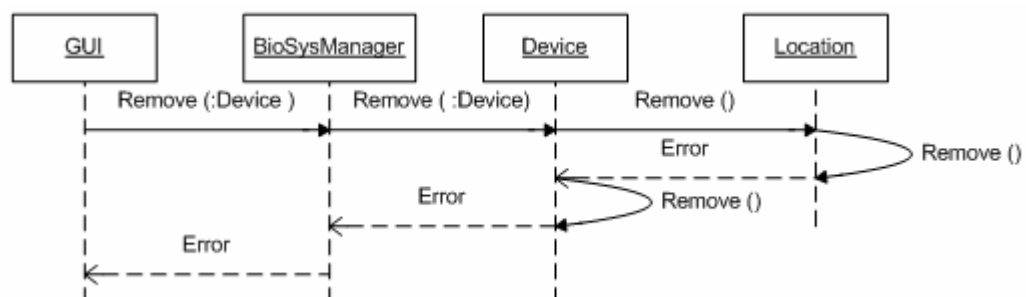
### Edit application

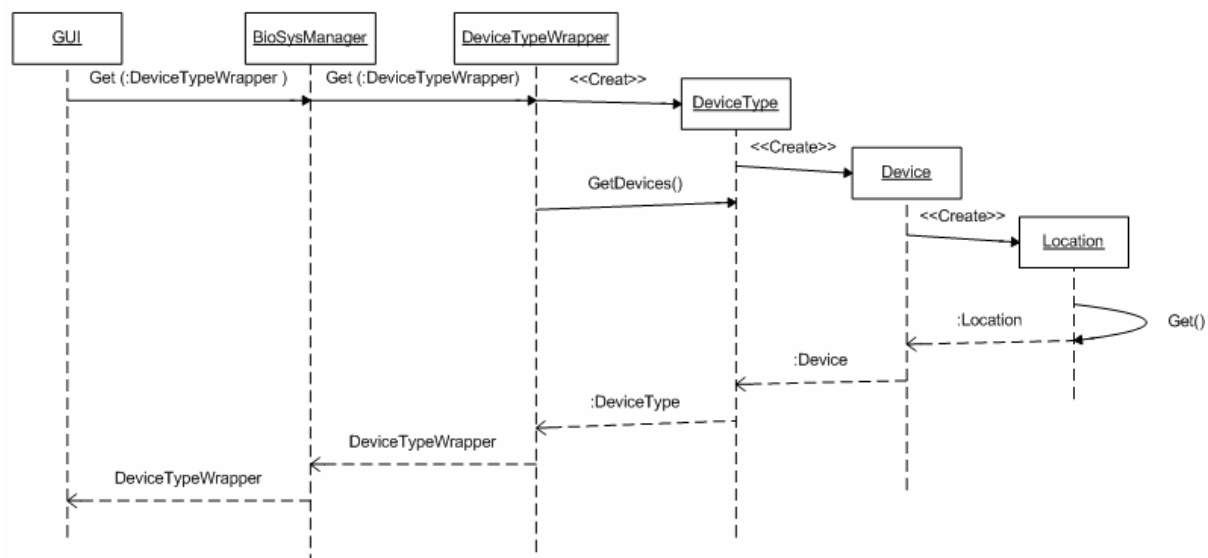
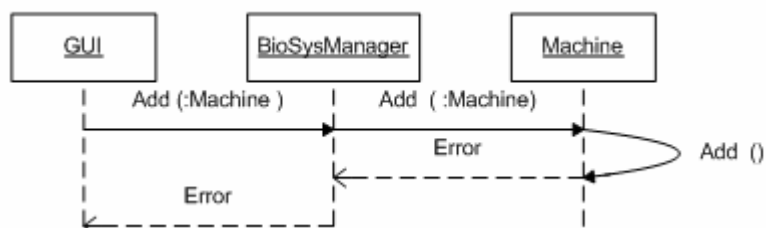
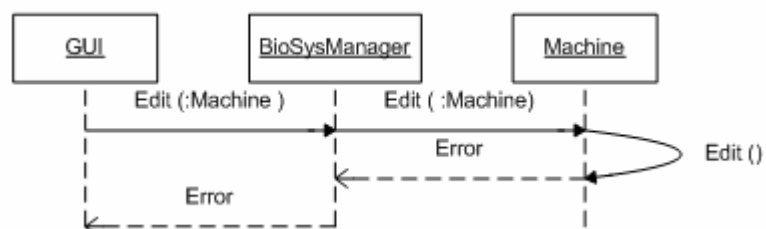
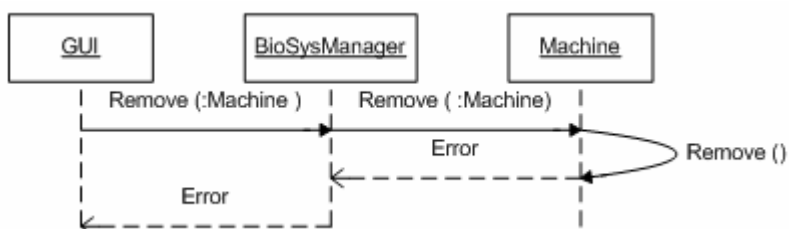


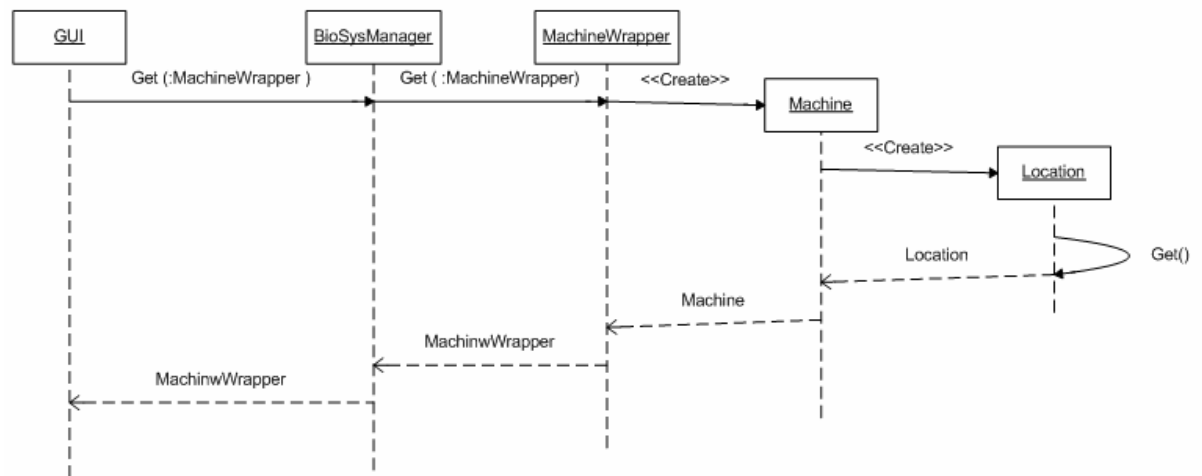
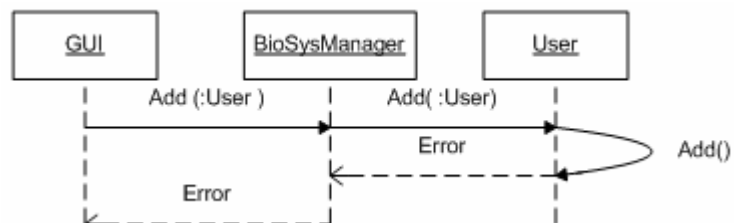
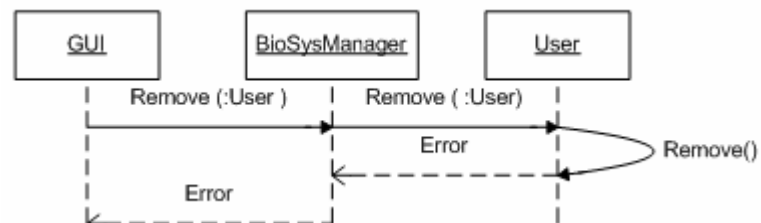
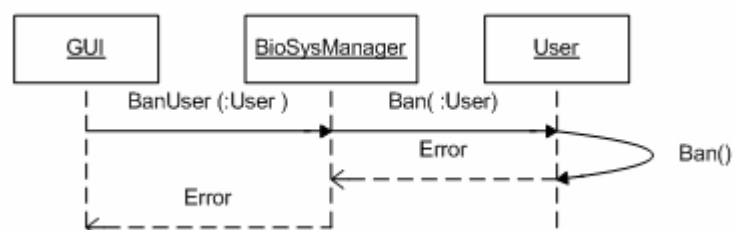
### Get all applications

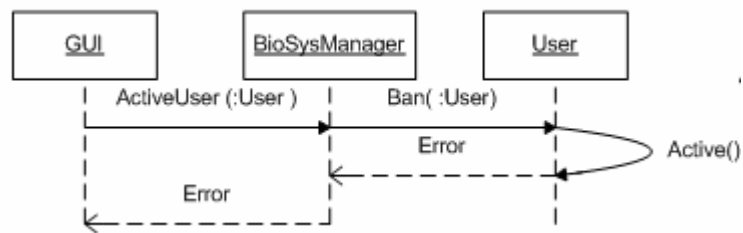
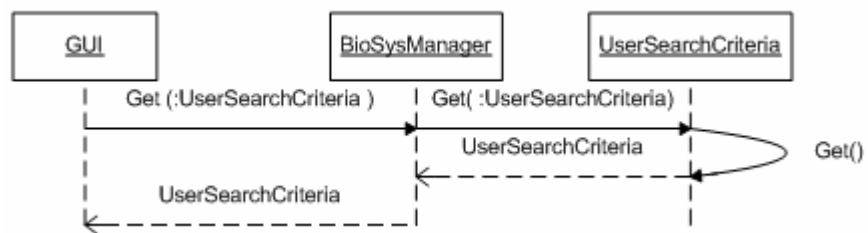


**Add device type****Edit device type****Remove device type****Get all device types**

**Add device****Edit device****Remove device**

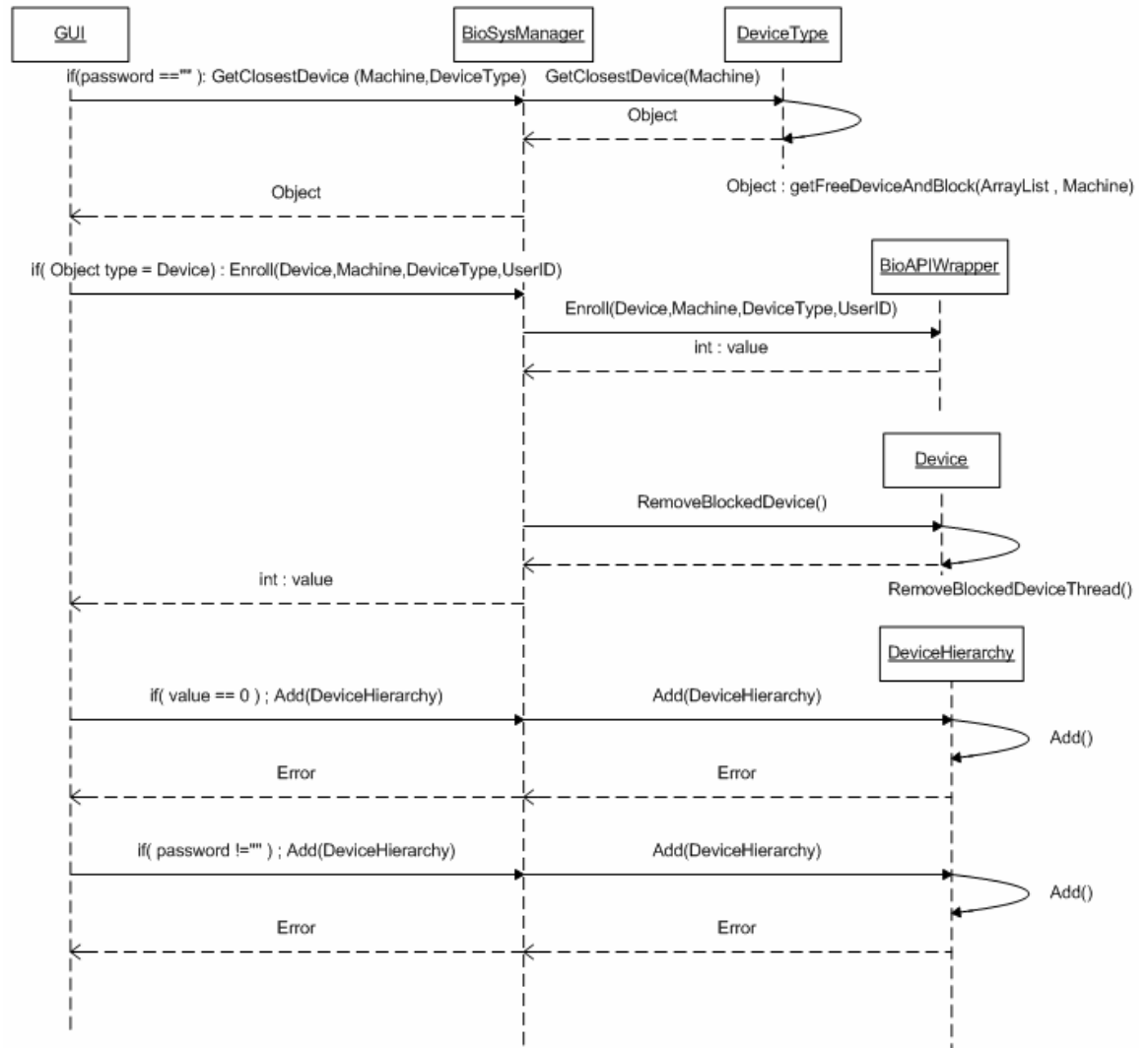
**Get all devices****Add machine****Edit machine****Remove machine**

**Get all machines****Add user****Remove user****Ban user**

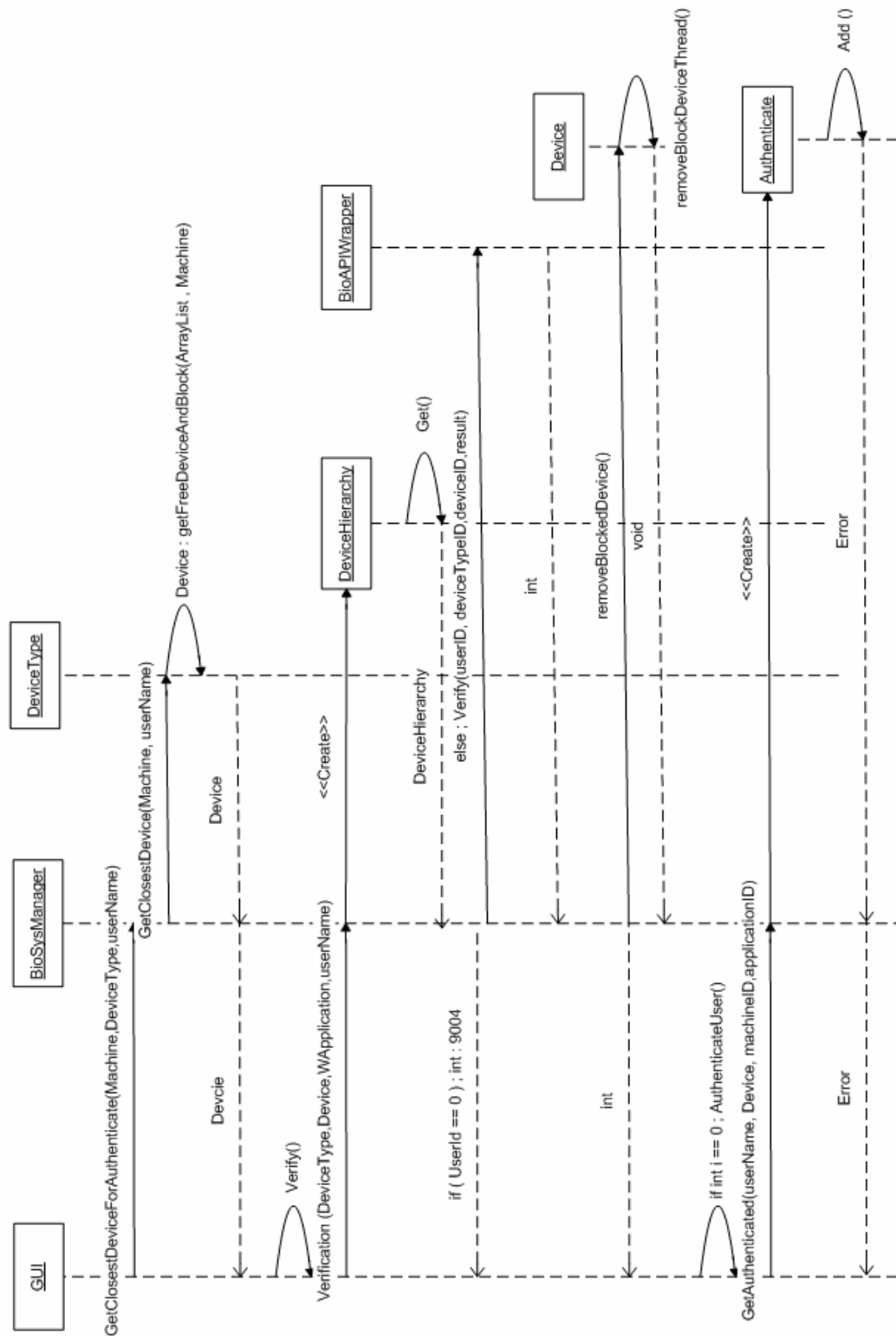
**Active user****User search**



## User enrol



## Verification



## Identification

