2000

Secure telemedicine system for home health care

Sridhar Vasudevan
West Virginia University

SECURE TELEMEDICINE SYSTEM FOR HOME HEALTH CARE

Sridhar Vasudevan

Thesis submitted to the College of Engineering and Mineral Resources
at
West Virginia University
in partial fulfillment of the requirements for
the degree of

Master of Science
in
Computer Science

V.Jagannathan, Ph.D., Chair
Sumitra Reddy, Ph.D.
James D. Mooney, Ph.D.

Department of Computer Science and Electrical Engineering
Morgantown, West Virginia
2000

Keywords: EJB, Java, Home Health Care, Telemedicine

SECURE TELEMEDICINE SYSTEM FOR HOME HEALTH CARE

Sridhar Vasudevan

ABSTRACT

This thesis describes a low-cost telemedicine system that provides home based patient care by linking patients with skilled nurses at the home care agency. The system employs compact vital signs sensors and a two-way real-time video conference over telephone lines. It stores the patient's medical records, still images and enforces clinical pathways during the televisits. Physicians, paramedics, and nurses can then have access to these records from anywhere, securely, through a Web browser.

This document discusses the underlying technologies, the features implemented in the prototype, and the methodologies used in developing the software. The prototype uses the Enterprise Java Bean [EJB] architecture and emphasizes security and scalability. Preliminary experience of its use is presented. A performance analysis of the system's behavior if it were scaled up has also been done.

ACKNOWLEDGEMENT

I sincerely thank Dr. Joe Cleetus for all his support during the two years of my research work at the Concurrent Engineering Research Center [CERC]. His guidance and patience at every step of the project has been immensely helpful.

I acknowledge with gratitude a large and continuing intellectual debt to my advisor Dr. Juggy Jagannathan. His expert guidance in middleware technologies has been very helpful.

I would also like to thank the other members of my committee Dr Sumitra Reddy and Dr. James Mooney for their support and review of several drafts of this thesis.

I am extremely grateful to the people at the Concurrent Engineering Research Center for having provided me with an oppurtunity to work with cutting edge technologies in the field of computer science.

-Sridhar Vasudevan

TABLE OF CONTENTS

TABLE OF FIGURES

1 INTRODUCTION

It's 9:30 AM Monday and Anna McCarthy is preparing to see her first patient at the home health care center. But Anna's patient is 23 miles away. Rather than drive there, Anna walks into the center's video room, because it's time to become a 'video nurse'.

"Good morning. This is Anna McCarthy. Are you ready for a visit?" Anna asks her patient over a regular phone line. The patient, 69-year-old Elwin Geyer, is ready. So, Anna asks the patient to turn on his TV set and video set-top box. Within a few seconds, the image of an emaciated patient suffering from congestive heart failure pops up on Anna's computer screen.

Elwin's wife, Jean, appears on screen with him. She lets Anna know that Elwin had an emergency on Saturday evening during which a nursing supervisor made a special video visit. "I see that," says Anna, glancing at Elwin's previous visit note. "There was shortness of breath. How's he doing now?" Elwin is apparently doing much better.

Anna conducts the same line of questioning she would if this were a face-to-face visit. "How is your appetite? Are you able to complete your daily exercise routines without discomfort?" After getting answers from Elwin and his wife, it's time to monitor the patient's blood pressure. So, Anna asks Jean to take her husband's blood pressure. Jean has learned to measure blood pressure with the digital unit provided. The computer screen shows the emaciated figure of Elwin, his wife wrapping the pressure cuff around him. Jean takes the blood pressure and holds the display in front of the camera. Anna types the reading into her visit note.

"How is the wound in your ankle healing up, can I take a look at it?".

Jean helps Elwin lift up the legs of his trousers. Anna presses a button on her telephone set and the image on her computer zooms into the wound. "Stay still, I am going to take a

picture". Anna freezes the image and stores it in her record. "It looks much better than it was last week" she comments after glancing at the picture she had taken the week before.

Are you coughing up anything?" The answer is no. "Do you feel congested in your chest, because it still sounds like you've got some congestion there." Jean explains that there is much less congestion than on Saturday when Elwin had a panic attack. Anna notes this on Elwin's visit record.

The video visit is part of a hectic schedule of home and tele-visits, which Anna will complete today.

This scenario illustrates the use of the system presented in this thesis to provide home health care to patients. Travel time and costs being a major issue, telemedicine systems help provide a quick and low-cost approach to providing care to home based patient

1.1 Problem Description

As the population ages, a great many people will stand in need of healthcare episodically. Whether the sojourn is long (chronic care) or short (a period of some weeks), it is expected that the cost of treatment at home will be much less than that of treatment in medical facilities. Rendering such service is achieved today by nurses from home care agencies going to visit patients at their homes several times a week, typically.

Home care has come a long way since the first home care agencies were established in the 1880s. Since then, growth has been phenomenal. About $36 billion was spent on home care

services in 1996 [1]. There were 18,874 home healthcare agencies and hospices in 1995, with about 60 percent of them Medicare-certified.[2]

In California, there were 14.3 million home care visits to 620,000 patients paid for by Medicare alone in 1994. And strong growth of 16 to 20 percent annually is projected to continue between now and the next 10 years [3]. There are about 1/2-billion home-health visits by nurses in the U.S. each year [4]. Kaiser Sacramento's Home Health Department alone has seen its new referrals increase from an average of 360/month in the first quarter of 1996 to 520/month in the first quarter of 1997 [5]. Many, if not most, of these require the cognitive and observational skills of the nurse, but not their actual on-site presence.

Certain factors, however, present problems. First, the terrain in rural hilly areas such as West Virginia is difficult. A lot of time is spent by a nurse traveling to and from patients who may live as much as 20 miles away from the home care agency. A nurse may not be able to see more than 2 or 3 patients in a day, though the effective time spent at the home of a patient may not exceed half an hour.

Secondly, cuts in budget in recent times, starting in 1998, have resulted in personnel layoffs at public home care agencies. Patients with less means have therefore been experiencing a curtailment or loss of service.

1.2 Telemedicine in home health care

Defined as using telecommunication and information technology to provide care at a distance from the care setting, telemedicine includes methods for evaluating, diagnosing,

treating, monitoring, and educating patients. Telemedicine makes it possible for a physician, nurse practitioner, clinical nurse specialist, or nurse case manager to monitor and supervise lay or professional caregivers.[6]

Most telemonitoring units are equipped with two-way, interactive television/computer screens, which clinicians and patients learn to integrate into day-to-day care. Information can be assessed in real time -- immediately as the information is received -- or in a store-and-forward format, such as when ECGs, X-rays, and video images are collected, stored in a personal computer, and forwarded to a consulting physician or other healthcare provider.

Telemedicine applied to home care (televisiting of a patient by a nurse remotely) holds the promise of eliminating the travel time of nurses to homes, thereby increasing the productive hours spent by a nurse.

In result, one could reduce the cost of a home visit and make it possible for nurses to service more patients at home than if they had to physically visit every patient. Of course, it may not be possible with today's technology to substitute with tele-visits more than that fraction of the normal visits in which the nurse does not require to perform physical care (bandaging a wound, performing an infusion, etc.)

Findings of Kaiser Home Health Department indicate the technology is dependable, and that average telehealth video visits are cost-effective and are about 60% shorter (18 minutes vs. 45 minutes) than on-site visits, with no decrease in patient satisfaction.[5]

1.3 Issues in health care systems

In any area of medicine, the quantum of data captured and recorded is very high. Every minute detail of the patient's past medical history is documented and stored. This set of data is frequently updated and viewed by various health care professionals - Nurses, physicians, paramedics etc. Sometimes, the data may be accessed simultaneously.

Patients may relocate or may need to be taken to a more sophisticated facility for treatment. The patient data must be accessible at the new location.

This leads to some key issues which arise in developing health care systems namely object granularity, security, support for access by multiple distributed clients, support for multiple types of clients, load balancing, etc.

1.4 Objectives

Most telemedicine systems which exist today require a high speed digital line for video conference. Also, expensive set of equipments need to be installed at the patient's residence in order to faciliitate telemedical home care.

After conducting an initial study on the clinical background of patients currently under home care, it was found that the average age of patients was sixty eight years. Many patients had chronic ailments or physical disabilities which prevented them from walking around and performing regular day to day activities. Considering the age and clinical condition of the patients, it was not reasonable to expect them to handle a computer. Patients lived in remote places and providing high speed digital lines for video conferencing was neither economically viable nor available in a timely manner upon demand from the telephone company. Hence the home care system has to operate under the following constraints:

- The cost of the equipment should be modest, in order that the delivered cost of telemedicine in poor under-served areas be actually lower than the cost of physically delivered care.
- The complexity of the equipment should be no more than what older patients, with impaired mobility and indifferent technical ability, can operate.
- Moreover, the equipment should not require more than the telecommunications infrastructure in remote rural areas can deliver on-demand.
- The quality of the telemedical care should be comparable to what is available by care *in situ*.

In this thesis we present a prototype of a home care telemedicine system which operates under the above constraints and focuses on the use of new developments in the field of distributed computing to allow various types of users to securely access and update patient data from different geographical locations. The work towards this thesis was done as a part of the *'Secure Collaborative Telemedicine'* project funded by the National Library of Medicine and the point of departure was the work done at Georgia Institute of Technology, Atlanta GA [7] and the University of Kansas Medical Center, Lenexa KS [8].

The goal of this task was to develop a low-cost telemedicine system that provides home based patient care by linking patients with skilled nurses at the home care agency. The system also had to provide facilities to store and update patient medical records, visit notes, capture and store still images from the televisit, provide secure remote access and generate reports.

The system must take into consideration the issues described in section 1.3. In particular it must provide access to multiple clients and multiple types of clients. Analysis of the distributed

computing system architecture in terms of its performance characteristics was also an important objective.To summarize, the main objectives in designing this system are:

Client / User Functionality

- To provide ease of use to all users
- To support use of system by multiple nurses
- To support queries
- To test the system with a few nurses actually treating patients and analyze the clinical aspects of the system.
- To facilitate capture of still images

Server Side Functionality

- To use a database management system for efficient organization of data
- To design a centralized server to which different types of authorized clients from various geographical locations could securely connect to gather and update patient data.
- To analyse the server under various load conditions and draw conclusions on performance characteristics

Security considerations

- To provide secure remote web based access for physicians to view patient progress
- To provide for security and confidentiality of patient data

1.5 Applicability and Usabilty

The system can be used to monitor vital signs and provide patient counselling on a regular basis. It provides a very simple interface to the patient comprising just a TV set and video

conference set-top box. So it can be used to treat patients of all ages, as long as their eye sight is reasonably good. It uses a regular telephone line hence patients located anywhere in the jurisdiction of the home care agency can be brought under the umbrella of telemedicine. Intrusive procedures such as administering an injection cannot be performed remotely, of course. A clinical pathway specific to the diagnosis has been implemented to guide the nurse through the tele-visit. Diagnosis-specific pathway modules for any ailment can be easily created and added to the system to facilitate use of the system to assist the nurse in providing care. The system provides for still-image capture and this helps monitor the progress of healing of wounds or any other physical manifestations of an ailment. As the system uses a central database, many nurses can treat many patients simultaneously. All data captured during the televisit is logged in the database which can then be viewed by any authorized client.

1.6 Methodology

The ***prototyping*** approach was used to design the system. Traditional approaches like the waterfall model or the life-cycle approach could not be used since all the requirements could not be found in advance without field-testing the system and gathering more requirements from users. Based on user feedback new features were implemented and the specification and design were modified accordingly.

The Object Oriented (OO) approach was used to develop the software as it provided a number of advantages over the traditional function-based approach. Applying a prototyping approach to function-based software would have been cumbersome; Moreover function-based software does not lend itself easily to modification. A number of standard 'Design Patterns' too were used in developing the software.

An Entity-Relationship (ER) diagram was used to design the database management system. The ER diagram was then mapped to the relational database tables.

1.7 Preview of Chapters.

The rest of the thesis is organized as follows: Chapter 2 discusses the literature review taken up for developing this prototype. Chapter 3 gives a functional overview of the prototype highlighting the various features. Chapter 4 discusses the detailed design of the system emphasizing the design patterns used. Chapter 5 describes the implementation issues. Chapter 6 discusses the results / analysis and presents the conclusion and future work.

2 LITERATURE REVIEW AND BACKGROUND

In recent years, systems have been developed to provide home health care through telemedicine. This chapter discusses the current state of development in this field. An insight into the various technologies used to develop the present home care system is also provided.

2.1 Existing Home Care Telemedicine Systems

Takano [9], discusses the use of videophones in providing telemedical care. It communicates through an ISDN line and has an integrated informatics system to provide electronic patient record management, graphical databases and other supporting information.

Burrow [7] presents a system that uses computer based monitoring stations placed in the homes of the patients. It also uses ISDN lines to provide communication.

Allen [8] provides an over view of a data abstraction instrument developed to monitor vital signs, make observations and provide counseling.

Kansas Care Inc, a company in Salina KS, markets a home care telemedicine system that provides cognitive evaluation, medication monitoring etc; it operates over regular telephone lines and provides for interactive audio-video visits.

Mahmud [10] describes a "plain old telephone system" (POTS) product for providing tele-care. It is organized around two modules: the home unit and the nursing unit. The nursing unit is set up at a central nursing station, and is staffed by a nurse who provides electronic house calls to multiple home units.

2.2 Drawbacks of existing systems

The systems described by Takano[9], Burrow[7] and Allen [8] use ISDN or other high speed digital lines. They have the following drawbacks:

- *Cost*

Using high speed digital lines is an expensive affair. If multiple patients are to be treated concurrently many digital lines will be required. The patients homes must also have high speed lines which greatly increases the cost.

- *Availability of digital lines*

Obtaining a digital line is a cumbersome process and usually takes anywhere between three to four weeks. This implies that a patient referred to home care cannot be televisited for a period of at least three weeks. In ailments like diabetes etc, the total duration of home care is about six to seven weeks; which means that more that for about 50% of the care duration, tele-visiting cannot be done.

Although Kansas Care and Mahmud[10] provide systems which use regular telephone lines, they suffer from the following drawbacks.

- *No integrated electronic patient record management*

There is no facility to electronically log visit details. Nurses use a separate software system to log the visit details. This requires the nurse to operate two separate systems at the same time.

- *Cannot capture still images*

Recording still images help monitor wounds and other manifestations of an ailment.

- *No facility for remote access*

If a physician wants to know the progress of his patient, he has to discuss with the nurse over phone. A remote access facility would not only obviate this mediation but also help paramedics learn about the patient's medical history without delay in case of an emergency.

- *No diagnosis specific clinical pathway*

A clinical pathway guides the nurse through the steps of a televisit. None of the current systems provide a clinical pathway; nor do they have a facility to add clinical pathways modules.

- *No automatic generation of Plan of Care*

The 'Plan of care' is a document created to finalize the treatment order and procedure for any home based patient. The contents of this document are partially recorded during enrollment of the patient in the home care agency and completed after the nurse's first visit to the patient. There is no facility to automatically generate the plan of care from the data distributed in the underlying database.

- *No Query*

If the nurse needs to access some specific information about a patient, she has to browse through all the pages of the patient record.

- *Inadequate Security*

Although all systems provide security and confidentiality of patient data from the outside world, there is no way to restrict access to patient data within the organization. A nurse has access, not only to records of patients under her care, but also to records of all other patients enrolled in the organization.

This thesis describes a system which overcomes the above drawbacks and provides a scalable and maintainable solution.

2.3 Background

This section gives a survey of the various technologies used in developing the software aspects of the system.

2.3.1 Multi-tier Design

The ability to partition applications into distributed tiers allows the flexibility of allowing each tier to do what it does best, instead of requiring "fat client" applications on the client systems. For example, middle tier servers can do the 'heavy lifting' of accessing remote databases, handling transactions, and aggregating information. This model replaces the two-tier client server model that required the maintenance of numerous drivers on the client systems to access enterprise applications.

A multi tier architecture consists of at least the following three components

Client-tier

Is responsible for the presentation of data, receiving user events and controlling the user interface. The actual business logic (e.g. generating "plan of care") has been moved to an application-server.

Application-server-tier
Business-objects that implement the business rules "live" in this layer, and are available to the client-tier. This level now forms the central key to solving 2-tier problems. This tier protects the data from direct access by the clients. Components on the server-side can be defined as configurable objects, which can be put together to form new application processes.

Data-server-tier
This tier is responsible for data storage. Besides the widespread relational database systems, existing legacy systems databases are often reused here.

It is important to note that boundaries between tiers are logical only. It is quite possible to run all three tiers on one and the same (physical) machine. The important point is that the system is neatly structured, and that there is a well-planned definition of the software boundaries between the different tiers.

A multi tier system overcomes the following drawbacks in the standard two tier model:

- Increased network load: since the actual processing of the data takes place on the remote client, the data has to be transported over the network. This leads to greatly increased network traffic.
- How to conduct transactions is controlled by the client. Advanced techniques like two-phase-commit can’t be run, since the transaction control is done by the client and multiple clients maybe connecting to the server at the same time.
- PCs are considered to be "untrusted" in terms of security, i.e. they are relatively easy to crack. Nevertheless, sensitive data is transferred to the PC.

- The import for change-management is drastic: because of ongoing changes in health care laws or procedures, processes have to be changed from time to time. Thus, possibly dozens of PC-programs will have to be adapted because the same logic has been implemented numerous times. It is then obvious that in turn each of these programs will have to undergo quality control, because all programs are expected to generate the same results again.
- The 2-tier-model implies a complicated software-distribution-procedure: as all of the application logic is executed on the PC, all the client machines have to be updated in case of a new release. This can be very expensive, complicated, prone to error, and time consuming.

Apart from overcoming the above advantages a multi-tier design provides a number of advantages, such as:

- Clear separation of user-interface-control and data presentation from application-logic. Through this separation more clients are able to have access to a wide variety of server applications. The two main advantages for client-applications are clear: quicker development through the reuse of pre-built business-logic components and a shorter test phase, because the server-components have already been tested.
- Re-definition of the storage strategy won't influence the clients. RDBMSs offer a certain independence from storage details for the clients. However, cases such as changing table attributes make it necessary to adapt the client's application. In the future, even radical changes, like let's say switching from an RDBMS to an OODBS, won't impact the client. In well-designed systems, the client still accesses data over a stable and user-friendly interface which encapsulates all the storage details.

- The three layers in general refer to the following: user interface layer, application logic management layer, physical data storage layer. The user interface layer can further me implemented as a thin client + servlet model or as fat client

2.3.2 Enterprise Java Beans

2.3.2.1 Overview

Enterprise JavaBeans (EJB) technology defines a model for the development and deployment of reusable Java server components. Components are pre-developed pieces of application code that can be assembled into working application systems. Java technology currently has a component model called JavaBeans, which supports reusable development components. The EJB architecture logically extends the JavaBeans component model to support server components.

Server components are reusable, prepackaged pieces of application functionality that are designed to run in an application server. They can be combined with other components to create customized application systems. Server components are similar to development components, but they are generally larger grained and more complete than development components. Enterprise JavaBeans components (enterprise beans) cannot be manipulated by a visual Java IDE in the same way that JavaBeans components can. Instead, they can be assembled and customized at deployment time using tools provided by an EJB-compliant Java application server.

[Since it is an ancient strategy to have server side procedures (called cataloged procedures or 'stored procedures' by Oracle) which can encapsulate a number of server-side procedures in a package, and which have the identical aim of resusing server side database-manipulation business logic, it is necessary to compare and contrast EJB with this long-standing approach]

2.3.2.2 Architectural Details

2.3.2.2.1 ENTERPRISE JAVABEANS SERVER

The EJB server provides an environment that supports the execution of applications developed using Enterprise JavaBeans technology. It manages and coordinates the allocation of resources to the applications.

2.3.2.2.2 EJB CONTAINER

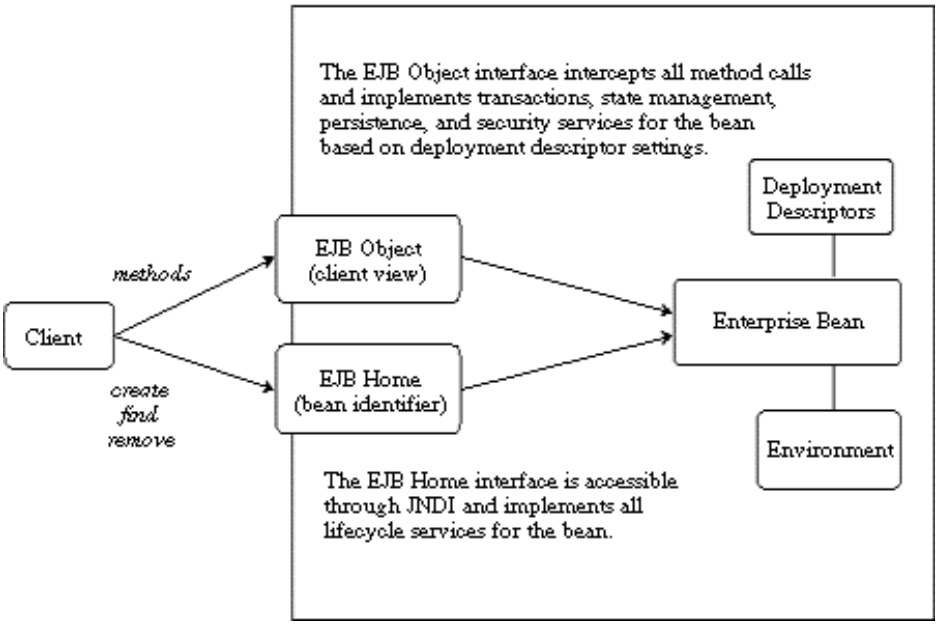

Fig 1. EJB Container

The EJB server must provide one or more EJB containers, which provide homes for the enterprise beans. An EJB container manages the enterprise beans contained within it. For each enterprise bean, the container is responsible for registering the object, providing a remote interface for the object, creating and destroying object instances, checking security for the object, managing the active state for the object, and coordinating distributed transactions. Optionally, the container can also manage all persistent data within the object.

2.3.2.3 Installing Enterprise Beans in an EJB Container

Any number of EJB classes can be installed in a single EJB container. A particular class of enterprise bean is assigned to one and only one EJB container, but a container may not necessarily represent a physical location. The physical manifestation of an EJB container is not defined in the Enterprise JavaBeans specification. An EJB container could be implemented as a physical entity, such as a multithreaded process within an EJB server. It also could be implemented as a logical entity that can be replicated and distributed across any number of systems and processes.

2.3.2.4 Transient and Persistent Objects

Enterprise JavaBeans technology supports both transient and persistent objects. A transient object is called a session bean, and a persistent object is called an entity bean.

2.3.2.4.1 SESSION BEANS

A session bean is created by a client and in most cases exists only for the duration of a single client/server session. A session bean performs operations on behalf of the client, such as accessing a database or performing calculations. Session beans can be transactional, but (normally) they are not recoverable following a system crash. Session beans can be stateless, or they can maintain conversational state across methods and transactions. The container manages the conversational state of a session bean if it needs to be evicted from memory. A session bean must manage its own persistent data

2.3.2.4.2 ENTITY BEANS

An entity bean is an object representation of persistent data that are maintained in a permanent data store, such as a database. A primary key identifies each instance of an entity

bean. Entity beans can be created either by inserting data directly into the database or by creating an object (using an object factory Create method ejbCreate). Entity beans are transactional, and they are recoverable following a system crash.

The EJB multitier application architecture provides for scalability, transaction management and method level access control

2.3.2.5 Comparison of EJB with other middleware technologies

Enterprise JavaBeans is a standard server-side component model for component transaction monitors. A component transaction monitor is a hybrid of traditional Transaction Processing [TP] monitors and Object Request Broker [ORB] technologies.

The Common Object Request Broker Architecture [CORBA] is an open standard ORB technology and provides a distributed object infrastructure for enterprise computing. EJB and CORBA compliment each other since EJB supports CORBA clients and EJB can be considered as a layer above CORBA which provides facilities for dealing with numerous server side issues such as threading & concurrency, access to data, access to legacy systems, security, authorization, server process lifecycle (creation, initialization, removal of objects), transactions (including distributed and two-phase-commit) and resource management.

If CORBA was used without a server component model such as EJB, the server developer would be forced to play two roles: that of an application programmer, and that of a systems programmer. A server component model allows the developer to focus only on the application logic. Developing server components for the EJB model is easier since one is programming at a 'higher level'. The execution environment (the EJB container) can automatically integrate the

application component with naming services, transaction services, security services etc., Application developers need not write any database code - data can be moved from databases to instance variables of the application component transparently.

The only other popular server component model which is a strong competitor to the EJB model is the Microsoft Transaction Service [MTS].

2.3.2.5.1 EJB vs. MTS

MTS, based on the Component Object Model (COM) which is the *middleware component model* for Windows NT, is used for creating scalable, transactional, multi-user and secure enterprise-level server side components. MTS provides a surrogate for in-process server-side components. MTS can also be defined as a component-based programming model. An *MTS component* is a type of COM component that executes in the MTS run-time environment. MTS supports building enterprise applications using ready-made MTS components and allows you to "plug and work" with off-the shelf MTS components developed by component developers. Just as a COM component can be modelled on the basis of interfaces and their implementation, MTS enforces modelling based on the component's state and behavior. MTS, through the COM infrastructure, handles communication between components using DCOM. This allows MTS to expose its components to Windows applications from anywhere on the net or the web. As long as you are only running on Windows NT, MTS components can be deployed on top of existing transaction processing systems including traditional transaction processing monitors, web servers, database servers, application servers, etc. Legacy system integration with non-Windows systems and MTS is achieved using the COM Transaction Integrator (COM-TI) technology. Also, it is important to realize that MTS is a stateless component model, whose components are always packaged as an

in-proc DLL. Since they are composed of COM objects, MTS components can be implemented in a variety of different languages including C++, Java, Object Pascal (Delphi), Visual Basic and COBOL

EJB is a *middleware component model* for Java and CORBA and is a specification for creating server-side, scalable, transactional, multi-user and secure enterprise-level applications. It defines a set of specifications and a consistent component architecture framework for creating distributed n-tier middleware. It would be fair to call a bean written to EJB spec a Server Enterprise Bean. Most importantly, EJBs can be deployed on top of existing transaction processing systems including traditional transaction processing monitors, web servers, database servers, application servers, etc. Since these components are written using Java, EJBs containing the business logic are platform-independent and can be moved to a different, more scalable platform should the need arise. If one is hosting a mission-critical application and need to move the EJBs from one platform to the other, it can be done without any change in the business-logic code. This allows one to "plug and work" with off-the-shelf EJBs without having to develop them or have any knowledge of their inner workings. EJB provides both a stateless and a stateful model and are packaged as jar files. Since EJB is built on top of Java technology, EJB components can only be implemented using the Java Language.

Component Types

MTS components have the following characterestics:

- MTS components typically execute on behalf of a single client.
- MTS components may be transaction-aware.
- They can update data in an underlying database.

- They are relatively short-lived, since their lifetime is limited to that of a client call.
- They may be destroyed as soon as they are done processing a client request and the component either calls SetComplete()/SetAbort(), or the transaction that it's part of ends, or the client calls Release() on the component.
- MTS components do not represent data that has to be stored in a database.

EJB components are of two types - Session Beans and Entity Beans:

The characteristics of a ***Session Bean*** can be summarized as follows:

- Session EJBs execute on behalf of a single client. A Session Bean instance is an extension of the client that creates it.
- Session beans may be transaction-aware.
- They can update data in an underlying database.
- They are relatively short-lived, since their lifetime is limited to that of their client.
- They may be destroyed when the EJB server crashes. The client has to establish connection with a new Session Bean object to resume any computation.
- Session beans do not represent data that has to be stored in a database.

Every ***Entity Bean*** has the following characteristics:

- Entity beans can share access from multiple users.
- They can participate in transactions.
- Entity beans represent data in a domain model.
- They are persistent. They live as long as the data lives in the domain model.
- Entity beans can survive EJB server crashes. Any EJB server crash is always transparent to the client.

- Entity beans have a persistent object reference. The object reference encapsulates the Persistent key for this Bean.

EJB also supports the notion of a component Handle which can be retrieved from a component (session or entity), stored in a data store, and later retrieved and re-hydrated to re-establish the connection to the component.

State Management

MTS uses a stateless model to ensure server resource management. The component does not maintain any state across transactional boundaries and is required to call SetComplete() or SetAbort() as often as possible to release state. As soon as a component completes its work, MTS deactivates the component.

EJB uses both a stateless and a stateful model. Stateful session beans maintain state information about the connection to a client, but this may not necessarily be database state. Database state consistency is ensured for entity beans using state management algoritms.

The container has automatic state management algorithms which take care of component state management. The EJB model uses methods like ejbLoad(), ejbStore(), ejbActivate() and ejbPassivate() to handle state management in each EJB class.

Persistence Management

MTS components are not persistent, even though they may contain information that needs to be persisted. If the need arises (which may be extremely rare), the component developer needs to invent and implement his own persistence management algorithm in the component, allowing the component to read or write data to the database as needed.

EJB has both persistent (entity beans) and non-persistent (session beans) components. The EJB spec defines a persistence programming model using methods like ejbCreate(), ejbPostCreate(), ejbRemove(), ejbLoad(), ejbStore(), ejbActivate(), and ejbPassivate() to handle persistence management in each EJB.

Entity Beans are inherently persistent and are of two types. Bean Managed Persistence (BMP) Entity Beans where the persistence management is handled by the bean methods using JDBC or SQL calls directly. Container Managed Persistence (CMP) Entity Beans, where the container tools vendor implicitly and transparently implements persistence management for the bean.

Session beans are inherently non-persistent components. However, they can implement their own persistence management, if required. They can also use the optional SessionSynchronization interface to implement transactional synchronization using the afterBegin(), beforeCompletion(), and afterCompletion() notification methods to signal transaction demarcation points

Security

MTS provides authentication and authorization using NT security. Distributed security services are provided by the NT Lan Manager (NTLM) security protocol supported by both Windows and Windows NT. The administrator can thus use the GUI tools supplied, to graphically associate a component or method with a list of users that have rights to invoke that component or method.

In **EJB** the security rules for each EJBean are defined declaratively in a set of AccessControlEntry objects defined in the deployment descriptor. An AccessControlEntry object associates a method with a list of users that have rights to invoke a method. The EJB container uses the AccessControlEntry attribute to perform all security checks on behalf of the enterprise bean

Deployment

Both MTS and EJB models support Application Partitioning. Administrators can easily partition an application across multiple servers by deploying an application's components into several packages, with each package running on its own server. This improves system fault isolation, while increasing application performance and scalability.

MTS includes a component packaging service that manages the complicated logistics of integrating, installing, and deploying many components as a single application. In addition to this, using MTS packages, developers and administrators can easily isolate components so they operate in their own system process. This is called Process Isolation**.** This provides an additional level of failure isolation and data protection.

EJB components can be packaged as individual enterprise beans, as a collection of EJBs, or as a complete application system. EJB components are distributed as jar files. The jar file contains a manifest file outlining the contents of the file, the enterprise bean class files, the Deployment Descriptor objects, and optionally the Environment properties. Deployment Descriptors defined in XML, specify the runtime settings for the EJB. The settings can be set at application assembly or deployment time.

2.3.3 Dynamic Web Page Generation

2.3.3.1 Java Servlets

Servlets are server-side Java programs that provide a means of generating dynamic Web content. They are not standalone applications that can be executed from the command line; instead, they run within Web servers. Of course, in order to run servlets inside a Web server the server must have a Java Virtual Machine running within itself. Unlike applets, servlets are not

constrained by security restrictions. They have the capabilities of a full-fledged Java program and can access files for reading and writing, load classes, change system properties, etc. They are restricted only by the permissions set by the file system, just like other Java application programs.

From a networking standpoint, servlets are Java applications that reside on the server side of an HTTP server. Since servlets are not subject to artificial security constraints, they enable the developer to extend Java programming to the server side of the HTTP protocol.

2.3.3.2 CGI and Servlets

Servlets provide an alternative mechanism to CGI programs for generating dynamic data. CGI programs have existed for a while; they are stable and universally accepted. They are language-independent (although they are not platform-independent). They are fairly easy to write. The question that comes to mind is, why should one consider replacing CGI with servlets? The main advantages of servlets over CGI scripts are:

Performance

Servlets offer a substantial improvement in performance over CGI. Each CGI request on the same server results in the creation of a new process. On the other hand, a servlet can continue to run in the background after servicing a request. Also, CGI programs are not threaded. Servlets can use threading to process multiple requests efficiently, provided that the JVM embedded in the Web server offers thread support.

Platform Independence

CGI programs are platform-dependent. Servlets are Java classes and follow the "write once, run everywhere" principle. Therefore, they are truly portable across platforms.

State Information

CGI programs are stateless because they result in the creation of a new process each time a request is serviced. A servlet has memory of its state once it is loaded by the server. The JVM running on the Web server loads the servlet when it is called. The servlet does not have to be reloaded until it changes, and a modified servlet may be dynamically reloaded without restarting the server. Maintaining state information allows multiple servlets to share information

2.3.3.3 Java Server Pages

A JSP page is actually compiled into a servlet when it is called, so JSP technology is truly an extension of servlet technology. However, JSP pages make servlets much easier to write and maintain by changing the approach to creating servlets.

In a servlet, fixed page elements (such as text) are embedded in the application logic (for example, in an out.println statement). Between the statements and the other programming overhead (such as preambles and boilerplates), it can be difficult to even find the fixed elements, much less change them.

By contrast, in a JSP page the application logic is embedded within standard HTML (or XML) templates. A page created with JSP technology can, therefore, be thought of as an "inside-out" servlet. Page authors can create or edit page design elements or text without affecting application logic at all. Because the JSP format automatically takes care of the programming overhead, and

the JSP container automatically compiles the page when it is called, the process of creating and revising pages is greatly simplified over that of writing and compiling servlet programs.

The application logic, which is embedded in the JSP page, may be in one of the following forms:

- Java scriptlets: Script fragments that perform simple processing
- JSP tags that call external functions, for example, a USEBEAN tag gives the page author easy access to a JavaBeans component that may perform some basic functionality
- Customized JSP tags; developers can create customized tag libraries that page authors may call with a simple, XML-like syntax

2.3.4 Security

While providing remote access to confidential patient data, a system must meet the following security requirements:

Authentication. Each party involved in the transaction should establish the identity of the others involved in the transaction. Authentication is usually provided through digital signatures and certificates.

Confidentiality. Confidentiality is an essential component in user privacy, as well as in the protection of sensitive information. Confidentiality in communications means that all the information that flows through the Internet should be intelligible only to the parties involved in the transaction.

Data integrity and availibility. Data sent as a part of a transaction should not be modifiable in transit without its becoming evident to the recipient. Similarly, it should not be possible for an intruder to modify data while in storage without detection. In the healthcare scenario, if health

information is corrupted, healthcare providers may not take right decision and may be harmful or even lead to the death of the patient.

2.3.4.1 SSL based security

Secure Socket Layer (SSL) is a session layer security protocol proposed by Netscape Communications Corp., SSL uses a combination of public-key and secret-key cryptosystems to provide authentication, confidentiality, and data integrity.

The SSL protocol is composed of two layers: the SSL Record Protocol Layer and the SSL Handshake Protocol layer. The Handshake layer allows the client and server to authenticate each other and to negotiate on the specific encryption algorithms and cryptographic keys to be used in the session before the application protocol transmits or receives data. SSL, takes the message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a Message Authentication Code (MAC), encrypts and transmits the result. Received data is decrypted, verified, decompressed, and reassembled, then delivered to the higher layers.

2.3.4.2 Certificate and Directory Servers

A digital certificate comprises of a block of data that contains the public key[15] and an endorsement by a Certificate Authority. Certificate servers are used to issue, revoke and manage certificates. They maintain an internal repository of certificate data in a secure database. The database contains information about certificates such as the date and time of issuance, who issued it, when it was revoked, expiration date etc. A certificate server works with a directory server such as the Lightweight Directory Access Protocol (LDAP) servers. If a client has a

certificate and wants to check the status of a certificate, or wants the latest certificate revocation list (CRL), it queries the Certificate Server. If a client needs a certificate, it queries the Directory Server.

3 SYSTEM OVERVIEW

3.1 System Requirements

The goal of this thesis was to develop a low cost system that overcame most of the drawbacks in existing home tele-care solutions. The key requirements of the system are listed below.

- *Integrated two-way video conference*

The interactive audio-video conferencing system had to be a part of the software system. This obviated the nurse having to use two separate systems while attending to the patient. It also made capture of televisit information into the database easy.

- *Still image capture*

It must be possible to record still images during any televisit.

- *Scalability*

Provision must be made for use of the system by many nurses simultaneously.

- *Maintenance of patient medical records*

The patient's past medical history, treatment order, enrollment information and all details up until the discharge of the patient must stored.

- *Monitoring vital signs and logging visit notes*

Vital signs and all details pertaining to each visit must be recorded as the visit is being conducted and made easily accessible. It should also be possible to log actual physical visits by the nurse to the patient's residence.

- *Confidentiality of patient data*

The patient records must be accessible to authorized personnel only. A nurse should only be able to access records of patients under her care. This necessitates the presence of a System Administrator to assign controlled access privileges

- *Secure remote access*

Patient information must be securely accessible through a standard web browser.

- *Generate Plan of care*

The system must generate a plan-of-care document corresponding to the HCFA 485 form based on the enrollment and data recorded at the first visit.

3.2 Approach

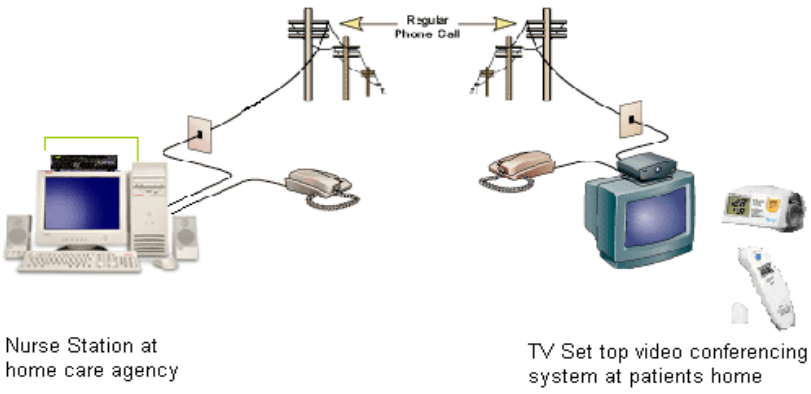

Fig 2. Approach

A Via TV VC1050 set top box, a Howard NCK41CV telecamera and a 13 inch color television set formed the video conferencing system at the patient site. At the home care agency, the nurse station computer was equipped with a TV tuner card, a set top box and a telecamera similar to the one at the patient's home. The TV tuner card enabled the nurse to view the patient in one

window on the computer screen, while she logged the televisit. The communication between the two sites was through a regular telephone line at 33.6 kbps(max). The video-conferencing system provided a resolution of 128 x 96 pixels at 15 frames per second.

3.3 Software Architecture

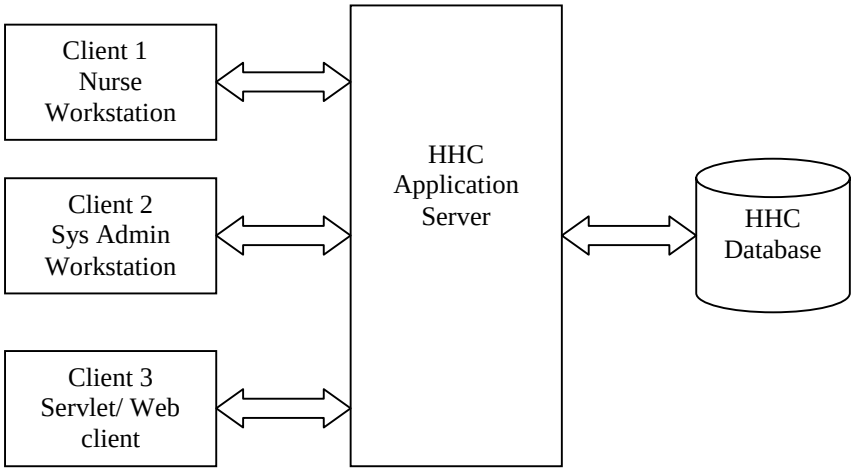

Fig 3. Software Architecture

The architecture consists of 3 tiers, a Client side application, a Middleware layer that implements the interfaces, and Server programs that mediate between the Client(s) and the database. The Client side of the system maybe the nurse application, the system administrator application, or any servlet client. The database stores all the patient records. The database records are mapped as entity beans in the Home Health Care (HHC) application server. Clients interact with the remote interfaces of these entity beans to update or add data to the HHC database. The HHC application server supports method-level access control through access control lists. It also supports method-level transaction management and isolation, and hence multiple clients can connect

simultaneously to the server. Chapter 6 discusses the performance of the HHC application server under various load conditions.

3.4 Functional Overview of the Software System

The system provides for three levels of users namely Administrator, Nurse and Data Entry Operator. Each type of user has specific privileges. The software features can be divided into Login/Logout, Administrator functions, Nurse functions, Data Entry Operator functions, and Remote Access functions. The following sections describe each of these in detail.

3.4.1 Login /Logout

In order to use the system the client side application must be installed on the users workstation. The client side application is a single common application for all types of users. To begin using the system, a user must enter a user name and password. All users must have a unique user name, irrespective of their type.

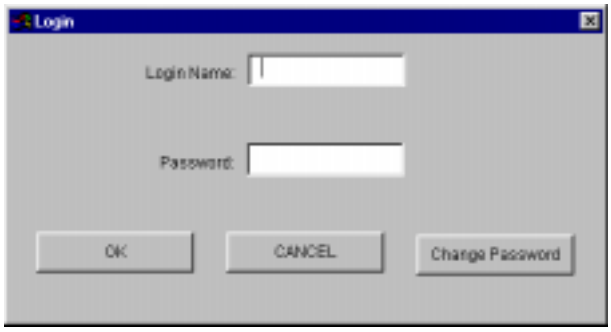

Fig 4. Login screen

Based on the username entered the appropriate user screen is loaded, ie., when the user enters his username/password, if authorization succeeds, the system detects the user type and loads the appropriate screen.

3.4.2 Administrator functions

A Nurse or Data Entry operator who wishes to use the system has to be provided with an account with appropriate privileges. The main function of the administrator is to add new users, modify the particulars of existing users and delete users from the system. The administrator is the only person with the ability to enroll users. He assigns login names and initial passwords. When an administrator logs in the system brings up the screen shown below.

Fig 5. Administrator Portal

Clicking on the desired icon gives the administrator a choice of adding, removing or modifying user information.

3.4.3 Nurse Functions

The Nurse is the most common type of user of the system. She has privileges to access and modify all of her patients data. The nurse is the only person who can schedule a televisit with the patient. She can also view the patient's plan of care. When a nurse logs in, the system brings up the following screen

Fig 6. Nurse Portal

The choice menu labeled "Select Patient" brings up the list of patients currently under the care of the nurse logged in. When the nurse logs in for the first time, this choice list is empty. The nurse would first have to add her patients into the system.

To add a new patient, the nurse clicks on the "Add New Patient" button which brings up a form into which she can enter the demographic details of the patient. On doing so, the new patient gets registered into the system.

To navigate through the various parts of a patient's medical record, the nurse selects the patient in the "Select Patient" menu and clicks on the "Patient Details" button. This brings up the screen shown below.

Fig 7. Patient Details

The complete patient record consists of four parts. The Medical Record, Visit Notes, Plan-of-care and Patient Demographics.

Medical Record. Clicking on the first icon (top left in Fig 7), brings up an iconic index of the patient's complete medical record as shown in Fig 8. The medical record consists of past history, treatment order, discharge plans, medications etc.,

Fig 8. Medical Record.

The nurse can update or view any aspect of the patient's medical record through this iconic index.

Visit Note. The nurse can view previous visit notes or create new ones by clicking on the 'Visit Schedule' button on the Nurse Portal (Fig 6) or by choosing the 'Visit Note' icon on the patient details screen (Fig 7). The visit note section begins with the screen shown below.

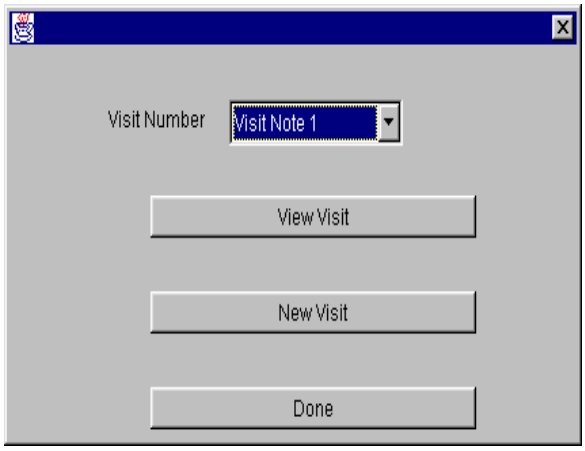

Fig 9. Visit Note Interface

There maybe cases when the nurse would need to make a fresh televisit when previous physical visits have not yet been entered into the system by the data entry operator. To take care of this situation the system gives the nurse the option of setting the visit number manually when creating a new visit. This facility is provided to accommodate physical face-to-face visits into the system.

Clinical Pathway. The method for conducting a visit is based on a 'Clinical Pathway' which provides a standard for care. A new televisit always begins with a clinical pathway interface. The pathway specifies the instructions and outlines the activities the nurse is expected to undertake on any particular visit. In the current version of this system the Clinical Pathway for 'Insulin-Dependent Diabetes Mellitus' has been incorporated. The pathway provides specific instructions on care to be rendered for the first seven visits. Subsequent visits record only the "Clinical Note". The instructions given and type of care rendered beyond seven visits is specific to a patient. This data can be recorded in the system in the "Remarks" or "Other Details" columns provided.

Based on the type of diagnosis entered into the system at enrollment, the appropriate clinical pathway is automatically loaded. Clinical pathway modules can easily be added into the system.

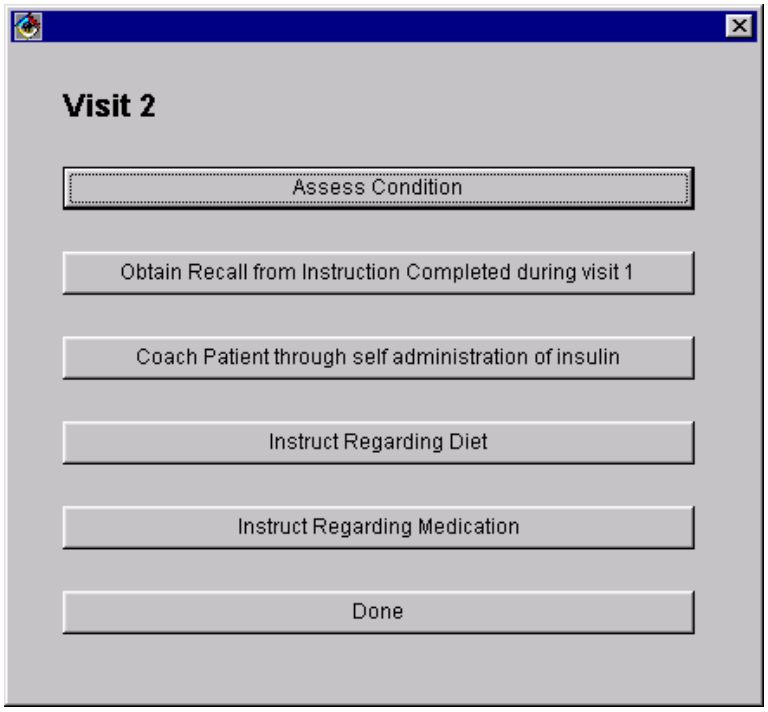

Fig 10. Clinical Pathway for Diabetes

The Assess Condition section consists of a set of screens which help the nurse assess the patient's current condition. For example, the screen to capture vital signs which is a part of assessing condition is shown below.

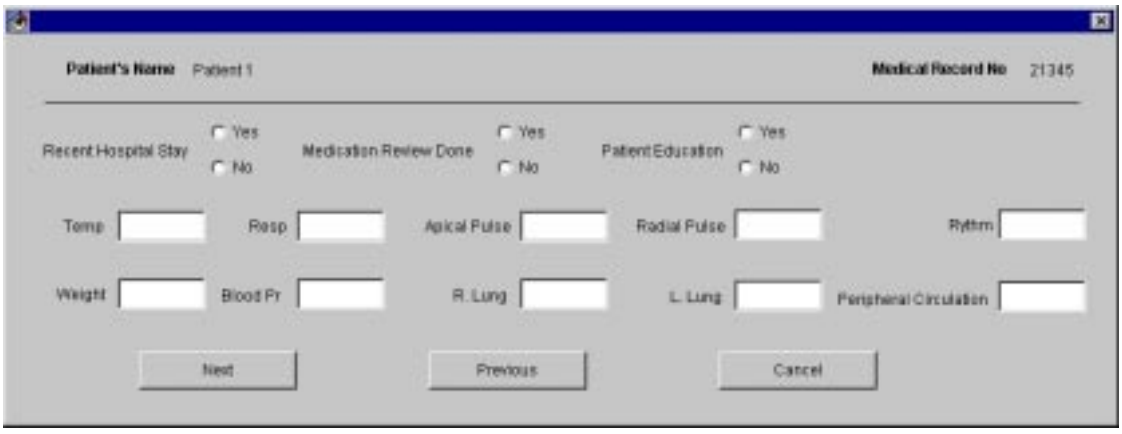

Fig 11. Vital Signs.

The remaining sections of the clinical pathway contain textual information which guide the nurse in coaching and counseling the patient.

Plan of Care. On clicking the 'Plan of care' icon in the patient details screen (Fig 7), the system automatically generates a Plan of Care document conforming to the HCFA 485 Home Health Certification and Plan of Care form. The data required for this document is gathered from the enrollment and first visit details present in the patient record.

Demographics. The demographics icon in Fig 7 leads the nurse to a screen which enables her to view/update patient demographic data such as: address, insurance provider, primary physician etc.

3.4.4 Data Entry Operator Functions

The function of this user is to add new patient records or update previous physical visits. The operations are similar to that of the nurse except that the data entry operator does not have permission to modify the patient records. When the data entry operator clicks OK on any screen, the information is committed to the database and he/she cannot make any further

modifications to the data. Any modification to be done has to be done by the nurse assigned to that patient.

3.4.5 Remote Access Function

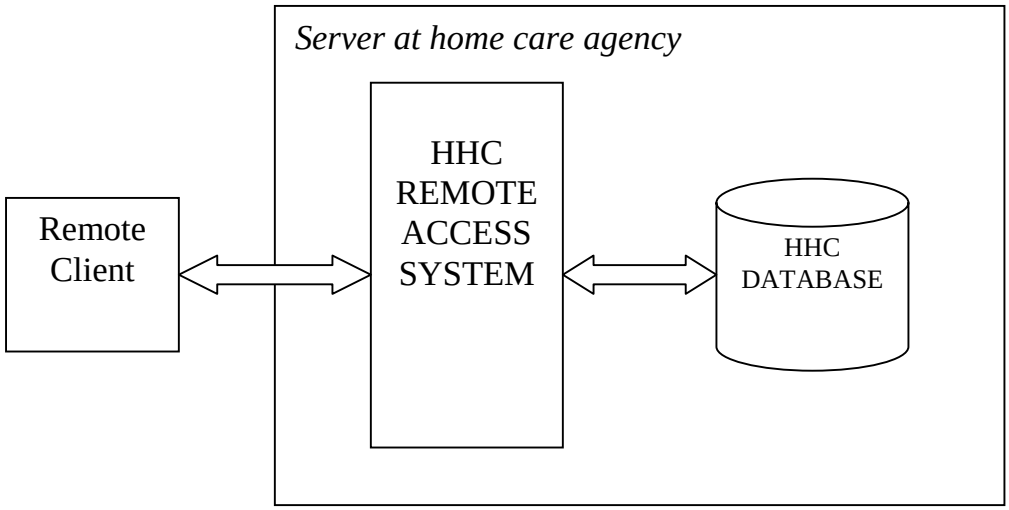

Fig 12. Remote Access System Architecture

A remote client can access the HHC System through any standard web browser that supports SSL. In order to obtain access to the patient data, the client must first obtain a digital certificate from the HHC Remote Access System. A client with a valid certificate can then access the system to obtain patient data. All communication is encrypted using SSL 3.0 and client authentication is mandatory. If the initial SSL handshake succeeds the user is prompted to enter his username and password as an additional security measure. This username/password is the same as the one issued by the administrator when the user is enrolled into the HHC System. If the user is authorized, he can access data pertaining to his patient.

Fig 13. Web Access Initial Screen.

In the current version, the following sets of data are accessible through a remote client

- List of patients under the user's care - User could be a nurse or physician
- All demographic information of any patient
- Patient vital signs recorded over all visits completed.

Fig 14. Patient Summary and Vital Signs

The HHC Remote access system is designed such that additional modules can easily be added and made accessible to the remote client without much difficulty. The design details of the remote access system are given in Chapter 4.

4 DESIGN

This chapter discusses the various issues involved in the design of the HHC System. The design considerations, methodology and constraints are described in detail.

4.1 System Architecture

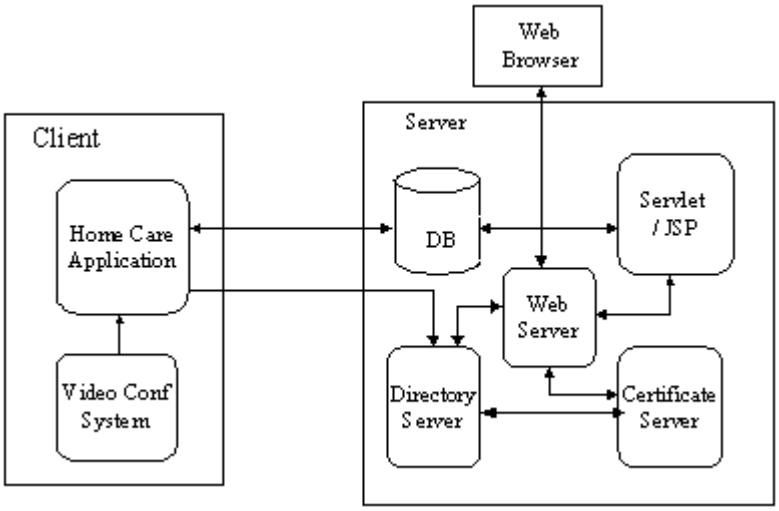

Fig 15. System Architecture.

The initial prototype of this system was developed using a two tier architecture as shown in Fig 15. The multi-tier design approach was applied to the visit note module of the system and is discussed in section 4.5. The nurse workstation hosts the client application and the video conferencing system. All data is stored in the database at the Server. A remote client can access data from the database through a Web server. The Certificate Server issues certificates to authorized remote clients. The Directory Server provides the necessary access control. The Servlet/JSP block facilitates generation of dynamic web pages for remote access.

4.2 Design of Database

The HHC system keeps track of the patient enrollment data, visit notes, plan of care, and the users who access these entities. The following sections discuss the requirements of the database, the ER diagram and the various database tables.

4.3 Requirements of the database

1. The database consists of a number of patients. Each patient has a unique id, demographic details, enrollment details, visit notes and a plan of care. Each patient is assigned to a nurse and is under the care of a physician.
2. Demographic details include name, address, insurance number, certification dates and date of birth.
3. Enrollment details consist of the data recorded during enrollment. This data is divided into 32 categories. Refer section 4.2.3 for details.
4. Visit notes include a unique visit note number, visit date and the data recorded during the visit. This data is divided into 13 categories. Refer section 4.2.3 for details.
5. The plan of care is a view created from the enrollment and visit note data.
6. The database has a list of users who can access the data. There are four types of users, namely, Administrator, Nurse, Physician, and Data Entry operator. The demographic details and passwords of each of these users is stored in the database. Each user has a unique login name.
7. The administrator adds, removes and manages the other users and assigns them login names and passwords.

8. A nurse can have many patients assigned to her. She can add, view and modify all data pertaining to patients under her care.
9. A physician can have many patients under his care. He can only view the records of patients under his care.
10. A data entry operator can only add or view enrollment and visit note data specific to a patient already enrolled in the system.

4.4 Entity-Relationship Diagram of the Database

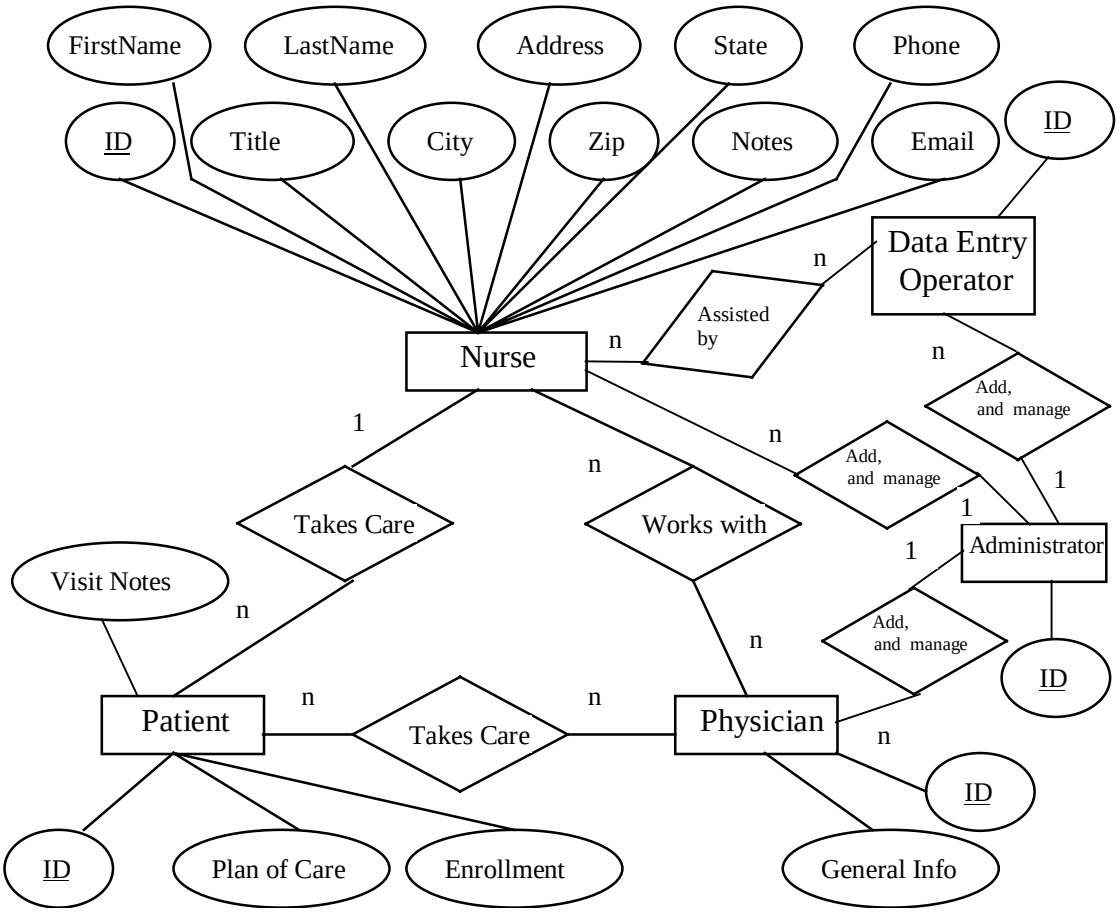

Fig 16. E-R DIAGRAM OF HHC DATABASE

Because of the size limitation of the picture, it is difficult to show all the attributes of entity Patient and entity Physician. Please refer section 4.2.3 for details

4.5 Relational Database Design

The relational database schema is derived from the conceptual schema using the Entity-Relationship model. Some of the tables are listed below.

Patient Table: This table keeps track of all patients enrolled in the home care agency.

```
Id                  int primary key,
first_name          varchar2(30) not null,
last_name           varchar2(30) not null,
address             varchar2(50),
city                varchar2(30),
state               varchar2(20),
zip                 varchar2(10),
phone               varchar2(20),
notes               varchar2(100),
nurse_id            int references nurse(id),
physician_id        int references physician(id),
medical_record_no int,
hi_claim_no         int,
sex                 varchar2(7) check(upper(sex)='MALE'
                    or upper(sex)='FEMALE'),
start_of_care       date,
certification_period_from date,
certification_period_to   date,
date_to_birth             date
```

Nurse Table: Maintains a list of all nurses in the home care agency

```
id                  int primary key,
first_name          varchar2(30) not null,
last_name           varchar2(30) not null,
title               varchar2(20),
address             varchar2(50),
city                varchar2(30),
state               varchar2(20),
zip                 varchar2(10),
phone               varchar2(20),
fax                 varchar2(15),
```

```
notes          varchar2(100),
email          varchar2(30),
loginname      varchar2(15)
```

Physician Table: Keeps track of all physicians who refer patients to the home care agency

```
id             int primary key,
first_name     varchar2(30) not null,
last_name      varchar2(30) not null,
title          varchar2(20),
address        varchar2(50),
city           varchar2(30),
state          varchar2(20),
zip            varchar2(10),
phone          varchar2(20),
fax            varchar2(15),
notes          varchar2(100),
email          varchar2(30)
```

User Data Table: Keeps track of all users of the system

```
id             int primary key,
name    varchar2(30) not null,
password       varchar2(30) ,
type           int,
nurseid        number
```

The enrollment module consisted of 32 categories and tables were designed for each category with the id in the patient table as foreign key.

Enrollment Category Tables

1. Background information
2. Vitals/allergies
3. Eyes
4. Ears
5. Nose
6. Throat
7. Mouth
8. Cardiovascular
9. Endocrine
10. Respiratory
11. Genitourinary
12. Genitalia

13. Gastrointestinal
14. Hematology
15. Neurological
16. Psychosocial
17. Pain
18. Skin
19. Musculoskeletal
20. Appliance/equipment
21. Living situation
22. Functional limitations
23. Activities permitted
24. Analysis of finding
25. Discharge plans
26. Skilled care
27. Activities permitted
28. Teaching/training
29. Medications
30. Standing orders
31. Diagnosis
32. Goals.

Similarly tables were created for the 13 visit note categories with the id in patient table as foreign key.

Visit note categories

1. Cardiovascular
2. Digestive
3. Emotional status
4. Genitourinary
5. Goals
6. Miscellaneous
7. Musculoskeletal
8. Nervous system
9. Other details
10. Respiratory
11. Skin
12. Travel data
13. Vital signs.

Plan of Care View : The plan of care was a view created from the various enrollment and visit note categories.

4.6 Design of Client Application

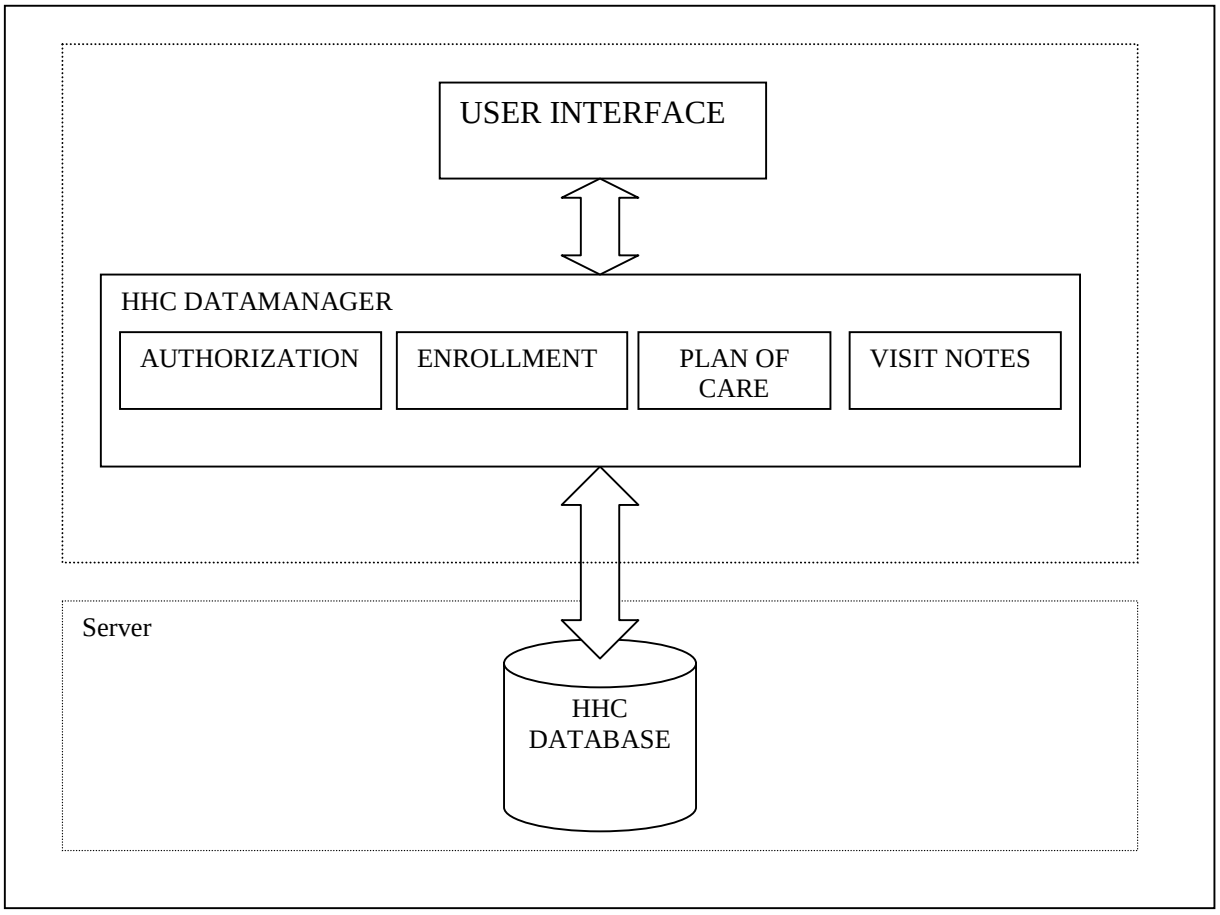

Fig 17. Client Application Architecture

The nurse uses the HHC System through the client application. The architecture of the client application is shown in Fig 17. Any modification to the database is done through a DataManager layer using the 'Chain of Responsibility' pattern [11]. Using the Chain of responsibility is appropriate since a number of classes attempt to modify the database. The technique to modify the database is standard and only the data varies from class to class. This approach provides a loose coupling between classes, and gives the added flexibility of being able to incorporate

additional features with ease. Any new class added only needs to transfer requests to the DataManager if it desires to access the database.

4.6.1 Login / Logout

The login procedure for all types of users is the same. The LoginDlg transfers the responsibility of performing the login procedure to the UserDataManager. If successful, the peformLogin procedure returns the type of user and the LoginDlg loads up the screen appropriate to that type of user.

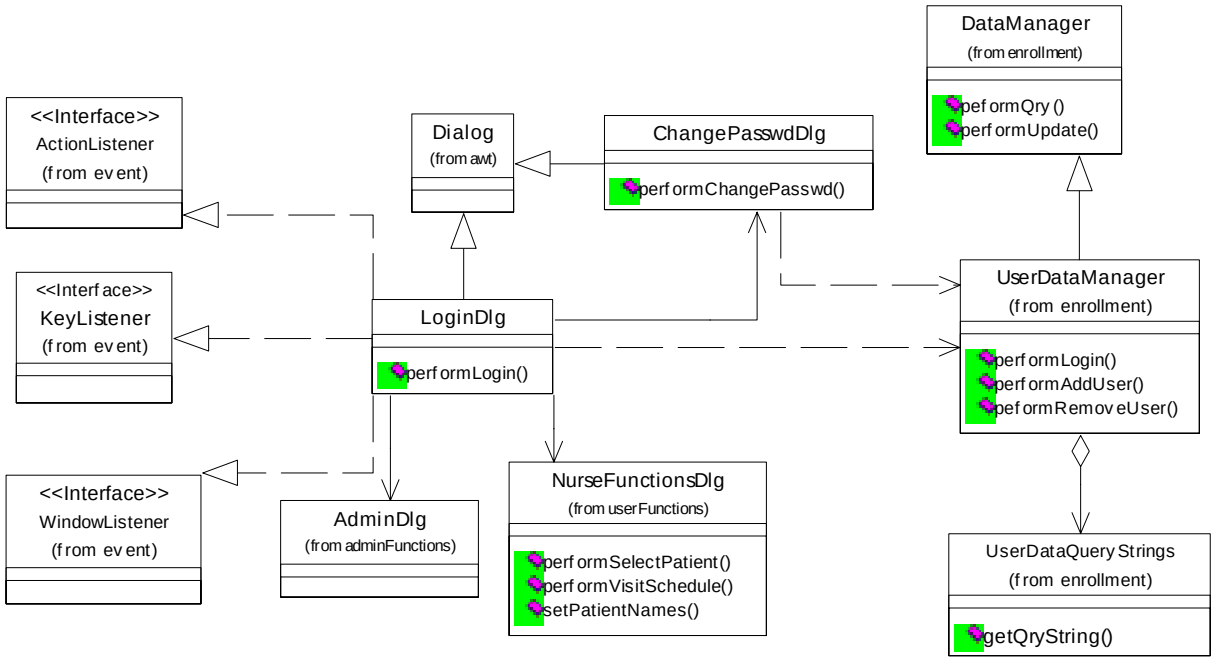

Fig 18. Design of Login System

The types of users, their privileges and their passwords are stored in the database as described in section 4.2.3. The 'DataManager' performs all database communication operations. It forms the base class for all DataManager classes in the HHC System. The LoginDlg also has an

association with the ChangePasswdDlg which enables the user to change passwords through the performChangePasswd operation.

4.6.2 Command Handling

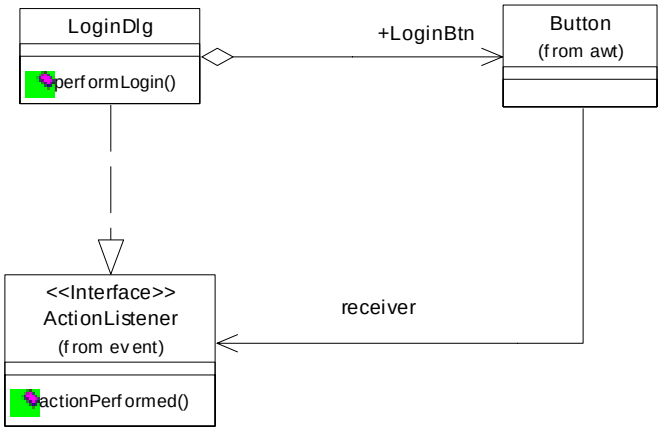

Fig 19. Command Handling

The design of the command handling feature is based on the Command Design Pattern [11]. The client for the event creates a separate command object or registers itself as the receiver and handler of the command. The command object is either a Mouse Listener object or ActionListener object. The receiver is a separate handler or the event client itself. The invoker object which is usually a button stores a reference to the command object. The invoker issues a request by calling either mouseClicked or actionPerformed on the command object.

Gamma [11] describes several advantages in using the Command Pattern.

- The object that invokes the operation is de-coupled from the object that knows to perform the operation.
- It is easy to add new commands without changing the existing classes.

4.6.3 Design of Administrator System.

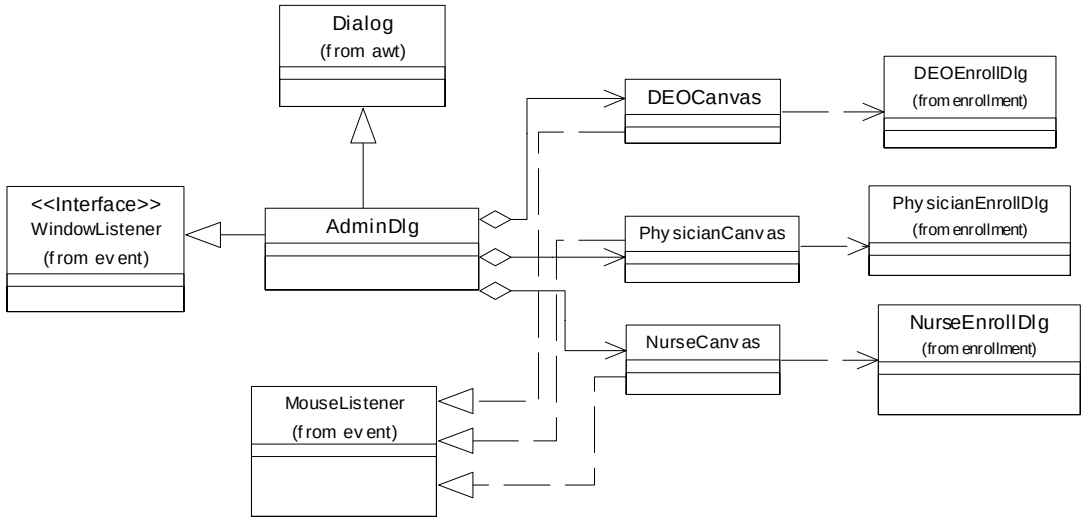

Fig 20. Design of Administrator System

The AdminDlg which renders the administrator portal screen is a base class which aggregates three handlers NurseCanvas, PhysicianCanvas and DEOCanvas which in turn are responsible for handling the administration of nurses, physicians and data entry operators. The AdminDlg renders the Canvas objects as components on the screen. When the administrator clicks on any canvas, the canvasClicked method on the appropriate Canvas component is invoked which in turn creates an appropriate EnrollDlg object , to handle enrollment. This design uses the chain of responsibility pattern [11] to transfer enrollment responsibilities to appropriate sub-modules. The main advantages are

- The admin portal contains separate components to handle administration of each type of user. Hence new components can easily be added at a later stage.
- Since the AdminDlg aggregates the administration components it automatically deletes all the sub-components when the AdminDlg is destroyed

- The Canvas components handle the mouse events independently. Hence it provides very loose coupling as no request is passed between the portal and the Canvas components.

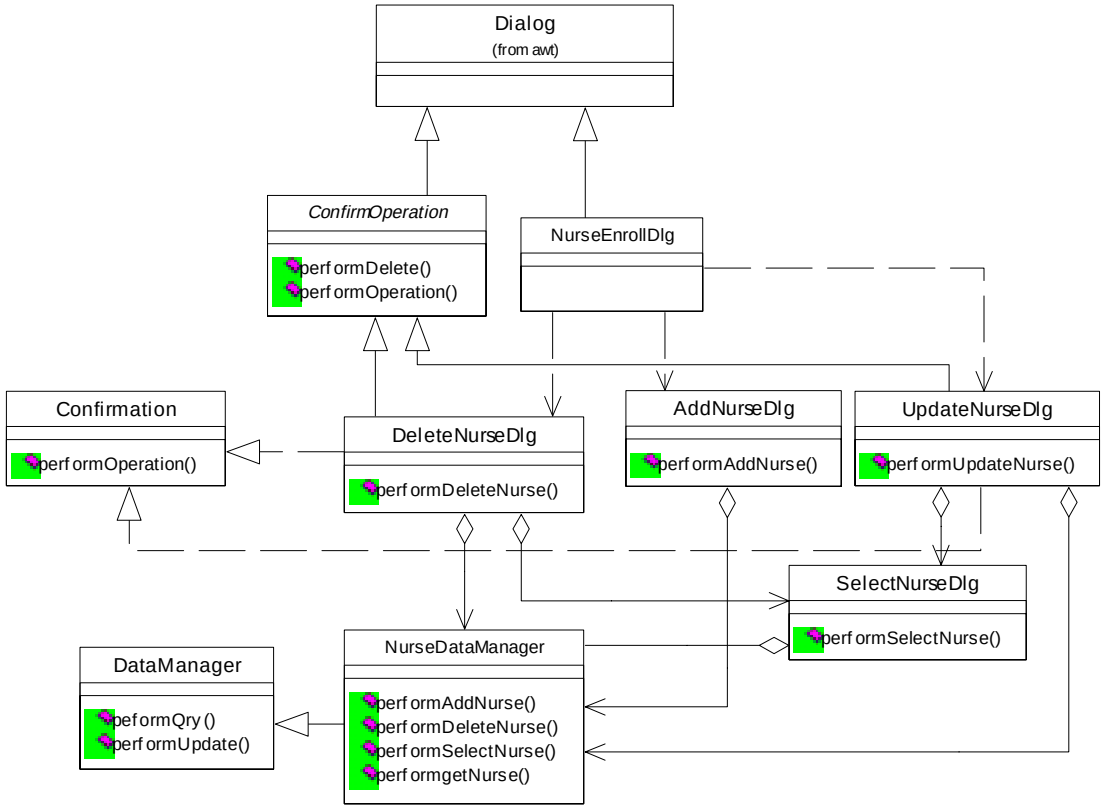

Fig 21. Design of Nurse Enrollment

Fig 21 shows the operations in nurse enrollment. A similar design is used for physician and data entry operator enrollment. In order to keep the diagram simple the fact that all the Dlg classes realize the WindowListener and ActionListener is not shown. The NurseEnrollDlg aggregates separate objects for adding updating and deleting nurses. These Dlg objects transfer responsibility of actual database communication to the NurseDataManager which specializes the DataManager class for nurse enrollment. The deletion and updating procedure go through a

confirmation interface which prompts the administrator to review and confirm the operation. The SelectNurseDlg enables the administrator to select the nurse for updating or deletion.

4.6.4 Design of Nurse System

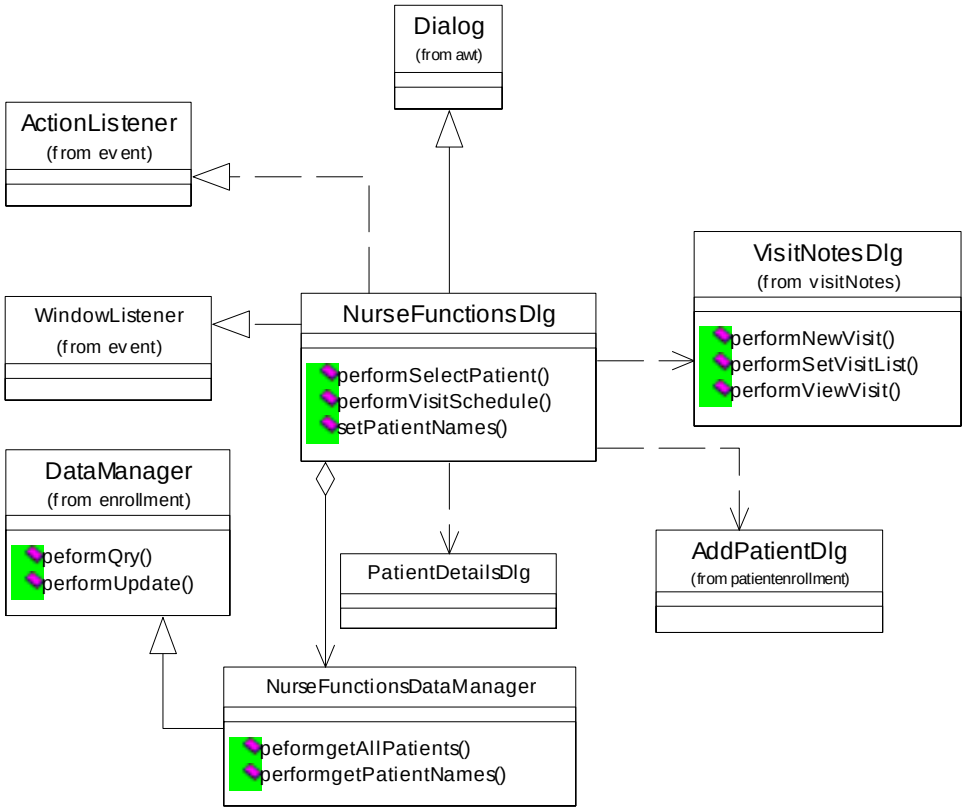

Fig 22. Design of Nurse Portal

When a nurse logs into the system, the LoginDlg creates a NurseFunctionsDlg object as described in section 4.3.1. The NurseFunctionsDlg is the nurse portal and acts as the base class for all operations to be performed by the nurse. The responsibilities of the NurseFunctionsDlg class are

- To display the list of patients under the care of the nurse
- Facilitate addition of new patients

- Access patient medical records, visit notes and plan of care data

NurseFunctionsDlg aggregates a NurseFunctionsDataManager to obtain the list of patients under the nurse. Other responsibilities are transferred in the following manner:

Addition of new patients - AddPatientDlg

Access to patient medical details - PatientDetailsDlg

Access to visit notes - VisitNotesDlg

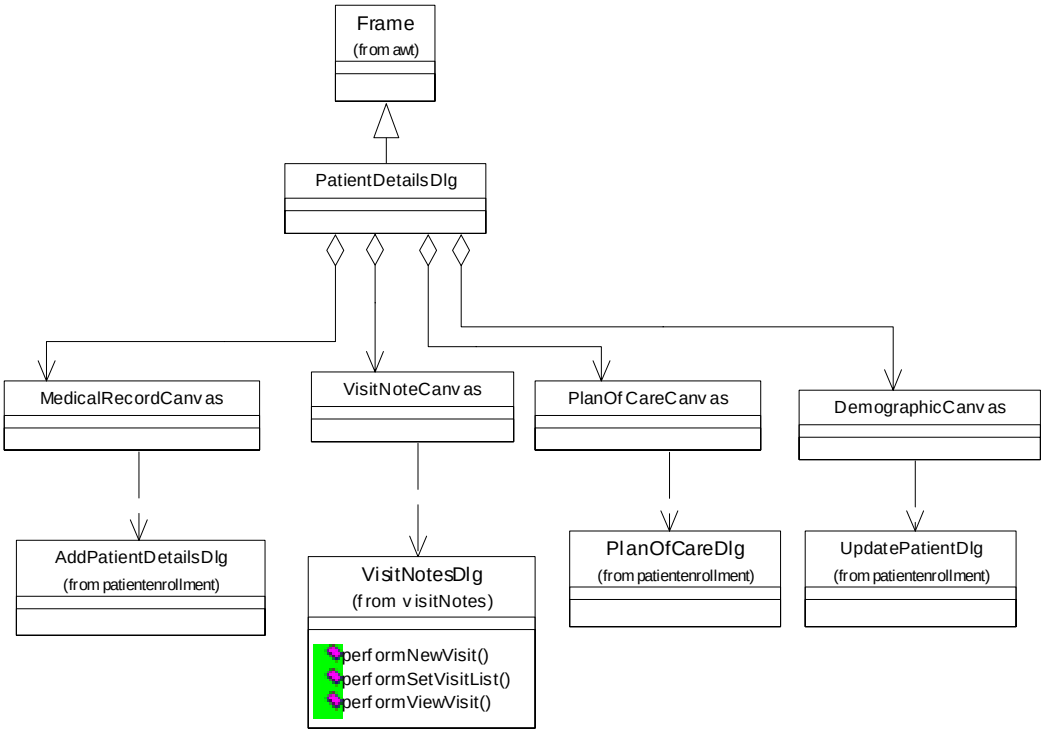

Fig 23. Design of Patient Details Screen

Fig 23 shows the design of the PatientDetailsDlg. It aggregates four Canvas objects as components namely MedicalRecordCanvas, VisitNoteCanvas, PlanOfCareCanvas, DemographicCanvas. Each of the canvas components and all the Dlg classes realize the WindowListener and ActionListener interfaces, although it is not shown in the figure. This

design is similar to the design described in section 4.3.3. Each canvas component captures the mouseClicked event to instantiate the appropriate Dlg classes.

Enrollment. The MedicalRecordCanvas instantiates AddPatientDetailsDlg [the iconic index shown in Fig.8] when a mouseClicked event is captured. The AddPatientDetailsDlg incorporates a similar design to provide a set of 32 components for modifying various aspects of the patient medical record. These components map directly to the enrollment tables described in section 4.2.3.

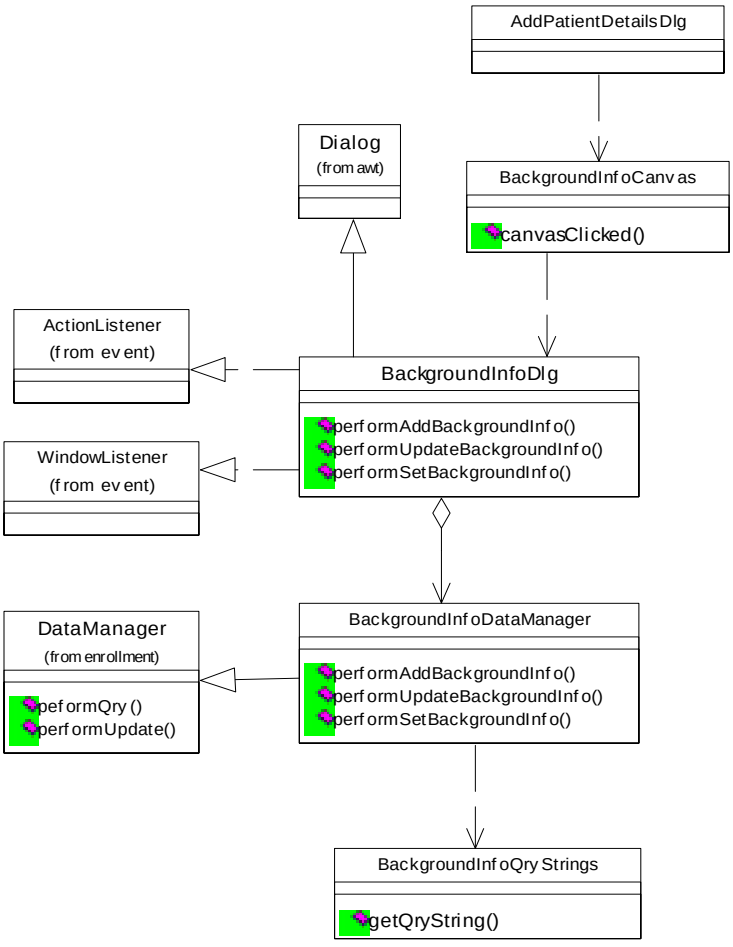

Fig 24. Design of BackgroundInfo Component

The design of one such component - BackgroundInfoDlg is shown in Fig 24. All other components embedded in AddPatientDetailsDlg use a similar design. The Dlg classes are instantiated only when a mouseClicked event results in the invocation of a canvasClicked operation on any of the components. This reduces the number of live object instances, thereby consuming less system resources. Also, the iconic index provides the nurse with random access to any particular sub-component. BackgroundInfoDlg transfers the responsibility of adding, updating or getting background information data to the BackgroundInfoDataManager which specializes the DataManager class for background information. DataManagers for each of the enrollment components are created. All these DataManagers inherit from the "DataManager" class which provides the methods for actual communication with the database. These DataManagers apart from providing get, set, and update operations specific to their respective components, also directly map to their respective tables in the database and provide accessor/mutator methods for each field in the table.

VisitNotes. The NurseFunctionsDlg and the PatientDetailsDlg both provide methods to access the visit notes module. The initial point for the visit note module is VisitNotesDlg. This singleton instance can either be instantiated from the NurseFunctionsDlg through the "Visit Schedule" button or the PatientDetailsDlg through the VisitNotesCanvas. The design of the visit notes module is described in Fig 25

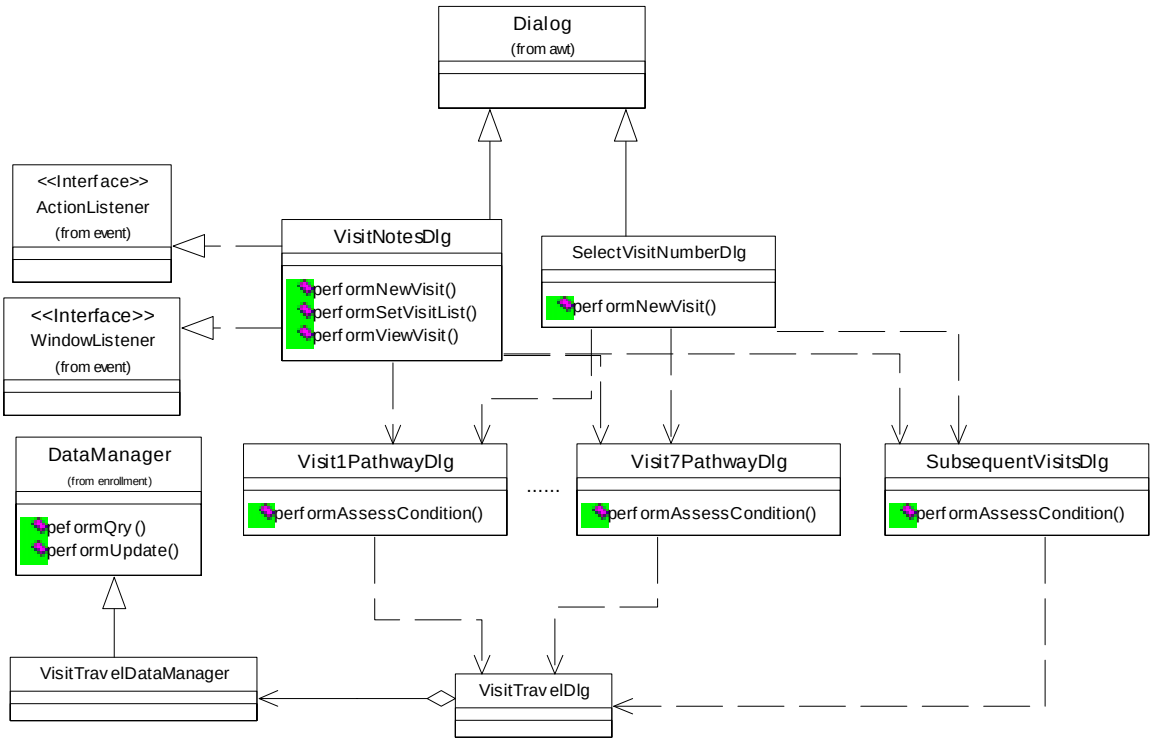

Fig 25. Design of Visit Notes

The VisitNotesDlg class has operations to create new visits and view existing visits. It also lists the previous visits notes logged in the system. The SelectVisitNumberDlg is used to change the default numbering of visits. This facilitates adding new televisits into the system when previous physical visits have not been recorded. The system implements a Clinical Pathway for diabetes. The pathway for each visit follows a separate set of procedures and each pathway is mapped to a Dlg class. However, for all visits the 'Assess Condition' step must be performed. Hence all the pathway classes invoke the VisitTravelDlg class when a mouseClick event is captured on 'Assess Condition' button. This is done through the performAssessCondition method. The VisitTravelDlg class is the starting point for assessing patient condition. Assessing Condition involves capturing information for the 12 visit note categories described in section 4.2.3. Each

assess condition class including VisitTravelDlg, provide a Next button which invokes an object of next category. The pathway specifies procedures upto seven visits. All subsequent visits follow a standard procedure which is specified in the SubsequentVisitsDlg.

Plan of Care. PatientDetailsDlg aggregates a plan of care component. This component invokes the PlanOfCareDlg. PlanOfCareDlg communicates with the database through a PlanOfCareDataManager just las the visit notes and enrollment module, to access the data in the plan of care view, which is a static view created in the database as explained in section 4.2.3

4.7 Design of Remote Access System

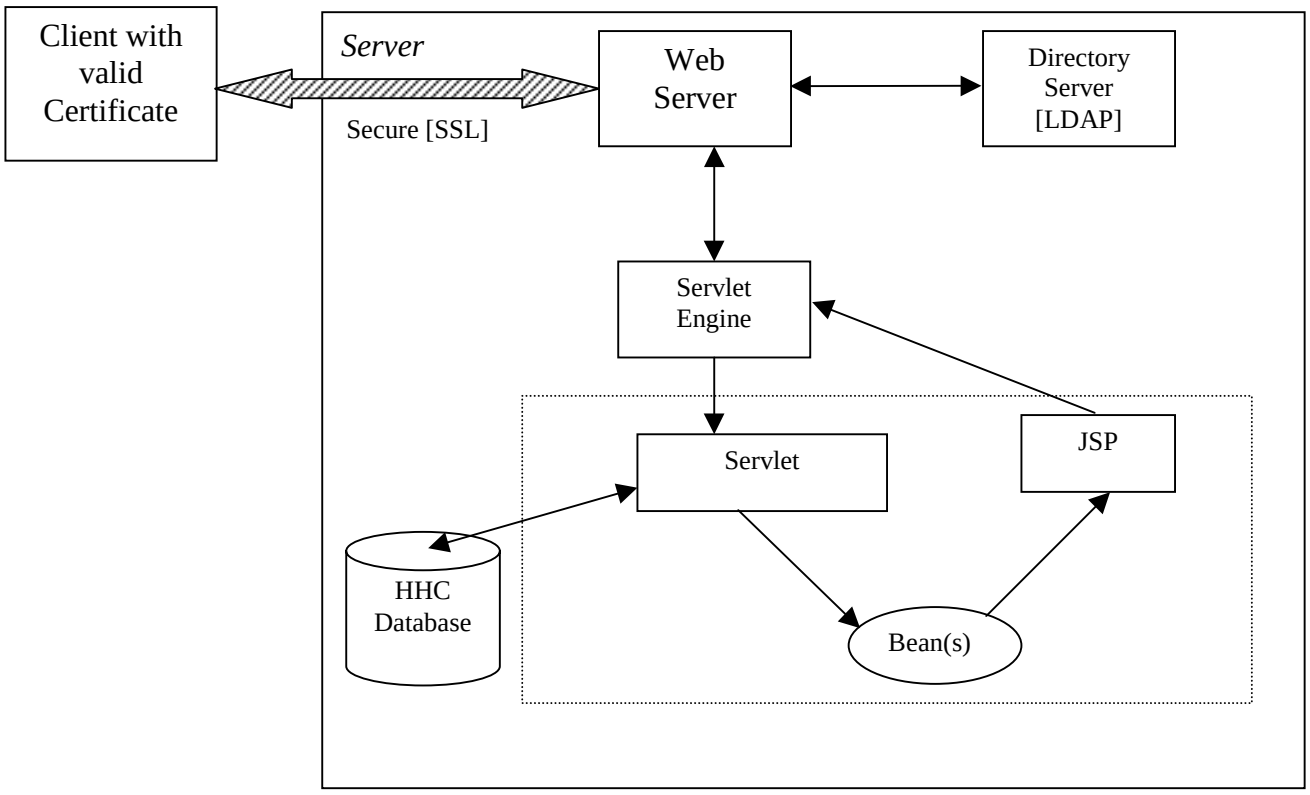

Fig 26. Remote Access System Architecture

Fig 26 shows the architecture of the remote access system. The pre-requisites for a client to access the system are

- The client must have a valid certificate issued by the HHC certificate server
- The client must be registered in the LDAP directory server
- The client must support SSL 3.0

The administrator can register a client in the LDAP directory server through the admin portal while creating the user, or by modifying the user profile. The client can obtain a certificate by connecting to the HHC certificate server and completing a 'request for certificate' form.

Security is provided using the SSL with client authentication. When a client connects to the HHC Web server, after the initial SSL handshake, the web server communicates with the directory server to ensure the user is registered in the HHC system and has been granted remote access privileges. If successful, the web server transfers further requests to the servlet engine which in turn invokes HhcStartupServlet. The servlet communicates with the database to obtain the required data and populates a corresponding bean. The request is forwarded to a Java Server Page, which then reads the data from the bean and forms the response HTML document. This document is sent to the client by the servlet engine through the web server.

This architecture provides the following benefits

- Separation of content from the logic. The logic to communicate with the database and obtain data is separated from the actual content. This greatly increases maintainability
- Peripheral data apart from what are accessed from the database, such as rendering information, font, display style, title etc, are separated from the code, and this can easily be modified through the JSP script file.

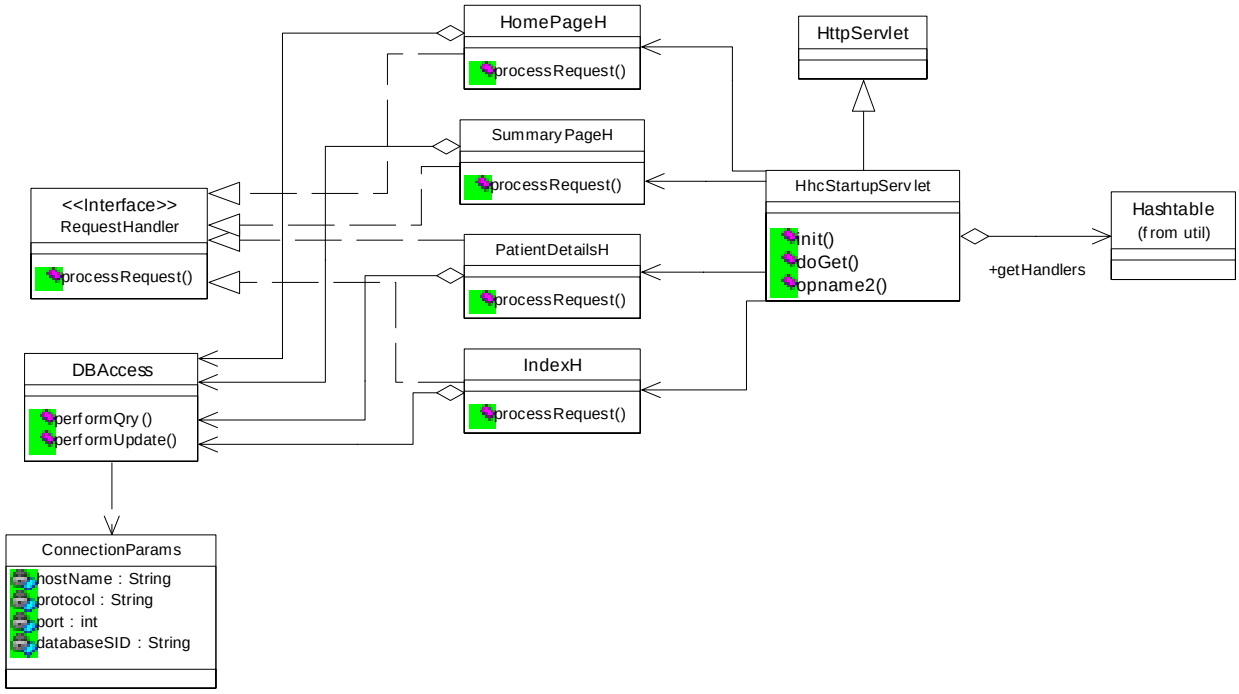

Fig 27. Servlet Design

The design of the servlet system is shown in Fig 27. In order to ensure security the entire web module has only one servlet HhcStartupServlet which provides a single entry point for all POST/GET requests in the servlet web application. This design uses the Delegation Event Model [12]. Any object which implements the RequestHandler interface can be registered to handle a specific event or request. If a RequestHandler is added to a servlet's GET request, the handler's processRequest() method will be called when the servlet receives this specific GET request. All request handlers must implement the RequestHandler interface. The RequestHandler has only one defined method, processRequest(), the first argument of which is an array of beans. The argument is an array to enable future additions. In the current version, the array will contain two beans - a user bean which indicates who is making the request and an error bean which indicates

how an error should be handled. The second and third arguments are the HttpServletRequest and HttpServletResponse objects respectively. The fourth is an array of service objects. In the current version, this array contains four elements namely the database connection, the servlet context, the request dispatcher, and the HttpSession objects. The handlers can use the services of these objects in order to handle the request.

Multiple sequential methods are not suitable to the multithreaded servlet environment, because instance variables are used to hold values. Hence, instance variables should be avoided whenever possible in servlets to keep concurrent request processing thread-safe.

The HhcStartupServlet takes all the GET requests and uses their request arguments to determine where to dispatch them. It instantiates all the specific request handlers and manages them so that they handle various requests correctly. The HhcStartupServlet maintains two RequestHandler collections. One for all GET requests and one for all POST requests. The collections are implemented with a Hashtable for quick and simple lookup. It can register a RequestHandler by the addGetRequestHandler() or the addPostRequestHandler() method with its given name as the key in the hashtable. This design requires that every HTTP GET request have the handler's name coded in the query string for all requests. Likewise, every HTTP POST request must have a hidden field named "handler" in the HTML form. Its value is the name of the requested handler. Whenever the HhcStartupServlet receives a POST request, it first gets the name of the handler, then finds the legitimate responder in its registered handler list, and finally delegates the handling.

In this arrangement, the HhcStartupServlet operates as a request-generating source and dispatcher, quite like an AWT component in JDK [13]. It can register event listeners and trigger

their handling procedures. Those objects that implement the RequestHandler interface are event listeners/handlers. They are responsible for handling their registered POST and GET requests.

In the current version, four request handlers are registered with the HhcStartupServlet namlely HomePageH, SummaryPageH, PatientDetailsH and IndexH. Their responsibilities are to fetch the appropriate data, populate a bean with the data and invoke the appropriate JSP page. The handlers use a dbAccess to communicate with the database.

HomePageH fetches the data required for the home page of the hhc web access application. The home page is the first page made visible to the client when connected. This data includes the name of the client (physician/nurse full name) and the full names of all patients under the care of this client.

When the client selects a particular patient, a GET request is sent to the SummaryPageH, PatientDetailsH, and IndexH which render themselves is separate frames. SummaryPageH obtains the patient summary which includes the patient and his referring physician's demographic information. PatientDetailsH obtains vital signs information. The IndexH lists the number of visits made by the nurse.

The beans populated by the requesthandlers save the data to be read by the JSP. They provide accessor/mutator methods for the data and also provide methods to obtain the data sequentially through an Enumeration. This enables easy access of data by the JSP page. Also the JSP page need not know the name of the fields while accessing them to create the HTML document. This enables easy modification and addition of fields to be displayed. If an additional field has to be displayed, slight modifications need to be made to the bean and the handler. The JSP page need not be modified.

Additional pages can be very easily added to the web access application by simply writing request handlers and registering them with the hhcStartupServlet.

4.8 Design of Enterprise JavaBeans

The visit notes and administration modules were designed and implemented using the Enterprise Java Bean architecture. This helped analyze the performance of a three tier system for home care application. The results of the analysis are discussed in Chapter 6.

The following sections describe the design of the EJB System for home care.

4.8.1 Design of Visit Notes

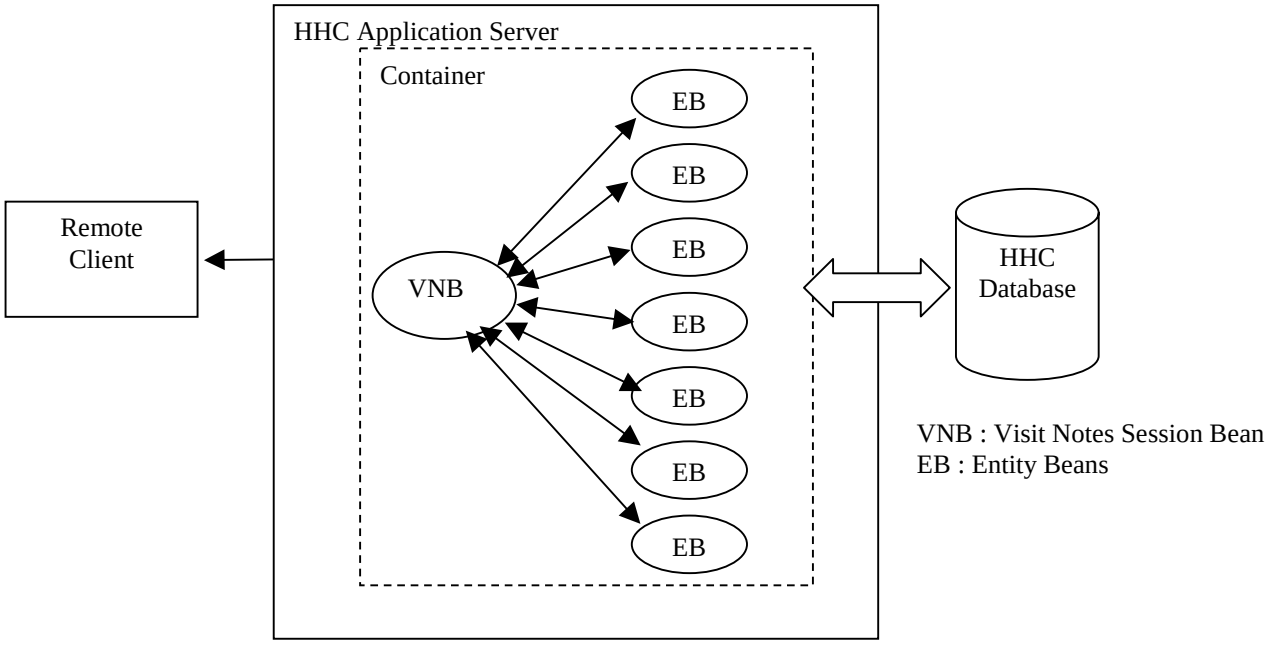

Fig 28. Basic architecture of HHC EJB System

The basic design of the server side beans for visit notes is shown in Fig 28. It implements the Façade Pattern [11]. The VisitNotesBean is a stateless session bean which acts as a Façade for a

set of Entity Beans. Each of these entity beans map to one of the visit note categories described in section 4.2.3. The entity beans use container managed persistence as each of them map directly to one table in the database. The use of the Facade pattern provides the following advantages.

- It shields clients from subsystem components (the entity beans), thereby reducing the number of objects that clients deal with and making the subsystem easier to use.
- It promotes weak coupling between the subsystem and its clients. Often the components in a subsystem are strongly coupled. Weak coupling lets you vary the components of the subsystem without affecting their clients. Facades help layer a system and manage the dependencies between objects. They can eliminate complex circular dependencies.

Clients communicate only with the VisitNotesBean which forwards the requests to the appropriate entity beans. The clients do not access the entity beans directly.

Clients can avail the following services provided by the VisitNotesBean:

- GET :This returns the complete visit note of the patient. The patient id is passed as an argument for the request.
- SET: The client can add a new visit note through this method
- UPDATE: The client can modify an existing visit note using the update method.

As mentioned earlier, the visit note module contains at least 10 categories and each category contains at least 10 fields. This means the complete object which needs to be marshalled across would contain over 100 fields. Managing such coarse grained objects is very cumbersome and

future modifications will not be easy. Hence the object to be marshalled to the client is also designed using the Façade pattern.

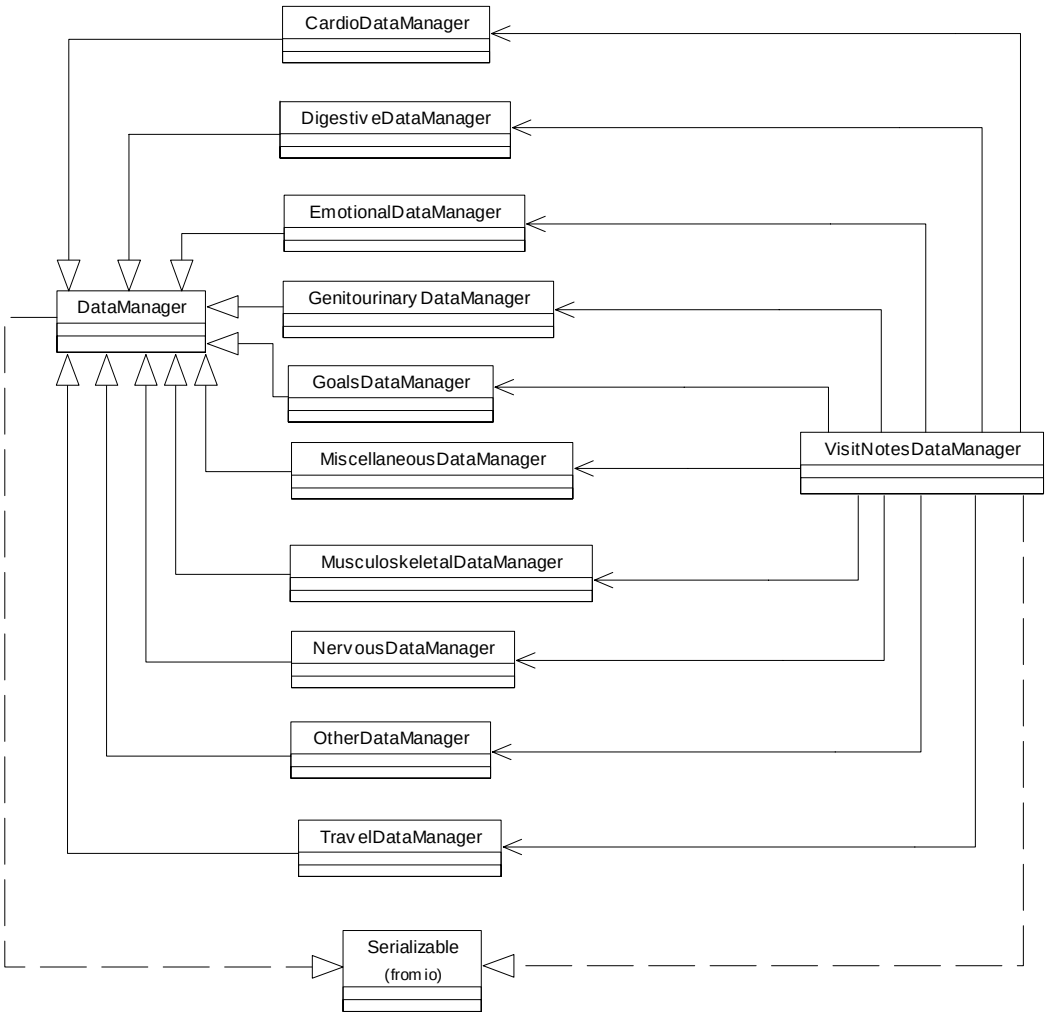

Fig 29. Design of VisitNotesDataManager

The VisitNotesDataManager acts as a Façade and holds the references of all the categories of the visit note. Each visit note category is mapped to a fine grained object and the VisitNotesDataManager holds a set of references of these objects. This considerably reduces object granularity. If the client uses only a subset of the categories, the remaining references in

the VisitNotesDataManager are set to null. This reduces the size of the object to be marshalled. Also, since only one object is marshalled and the client interacts with only one session bean, the network traffic is considerably reduced using this design.

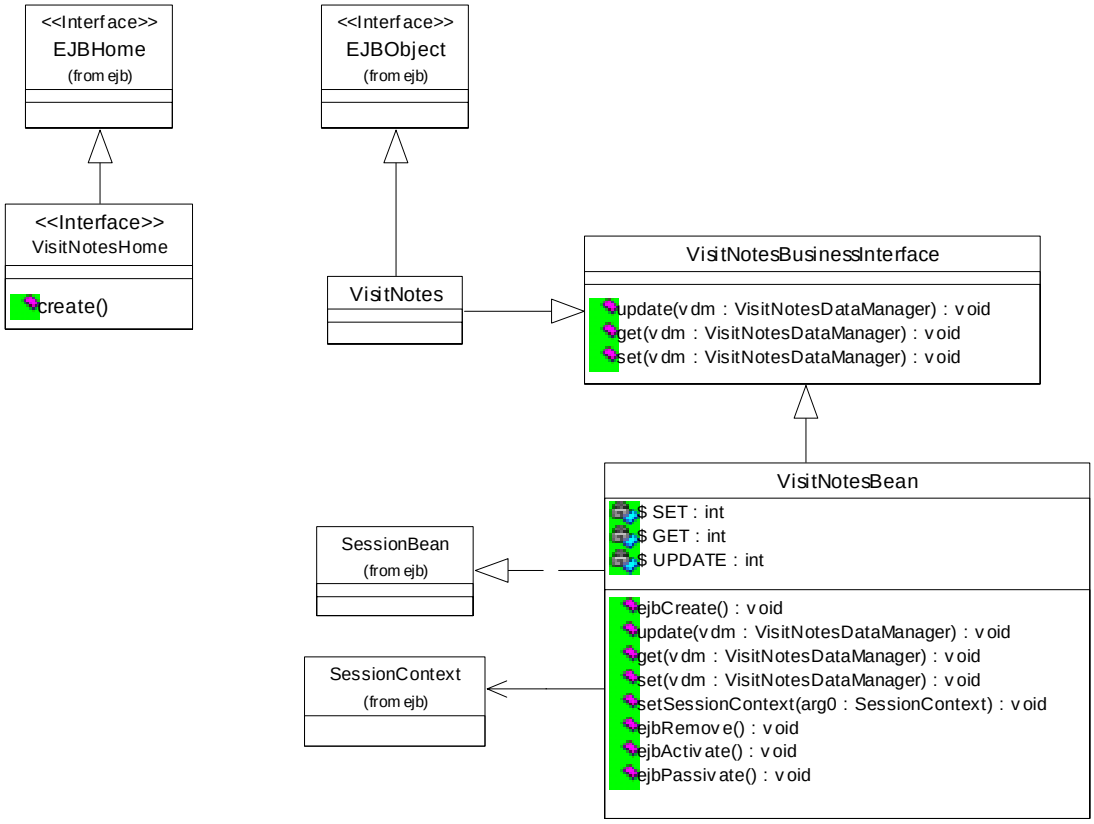

Fig 30. Design of VisitNotesBean

The design of the VisitNotesBean is shown in Fig 30. The design implements the Business interface pattern[14]. The EJB specification forces the bean class to provide an implementation for all methods declared in the remote interface. The spec also recommends against the bean class directly implementing the remote interface using the "implements" Java keyword.

This increases dependency on the EJB vendor whose EJB compiler must be used to enforce the implementation of all the bean class' remote interface methods. And the bean has to be compiled

twice: the first time with the standard JDK compiler to fix any errors it reveals, and a second time using the vendor's EJB compiler to fix any errors detected.

Many of the errors found by the second compilation involve methods declared in the remote interface that aren't declared correctly in the bean class. This could be caught in the first compilation step if the bean class were able to implement the remote interface directly. Then the Java compiler could use the standard rules for enforcing the implementation of an interface.

The workaround for this problem is to use the Business Interface pattern. This interface is defined just as the bean's normal remote interface; only it doesn't extend javax.ejb.EJBObject. The business interface declares all the methods that can be called on the business object, but doesn't make any assumptions about the underlying implementation.

The VisitNotesBusinessInterface defines the signatures of the Get, Set and Update methods. The VisitNotesBean is defined as a stateless session bean since it only acts as a façade and does not have to maintain state between invocations. Maintaining the VisitNotesBean as a stateless session bean greatly increases performance as multiple bean instances can be instantiated which are independent of each other. Also no passivation or activation needs to be done in a stateless session bean since it does not maintain state.

The implementation of the Set and Update methods accesses each of the references from the VisitNotesDataManager façade and passes the requests to the corresponding entity beans. The Get method creates entity beans for each category, calls the Get method on each of the entity beans to form a VisitNotesDataManager object which is returned to the client.

All the business interface methods of the VisitNotesBean communicate with a LogBean to log the time taken for each visit.

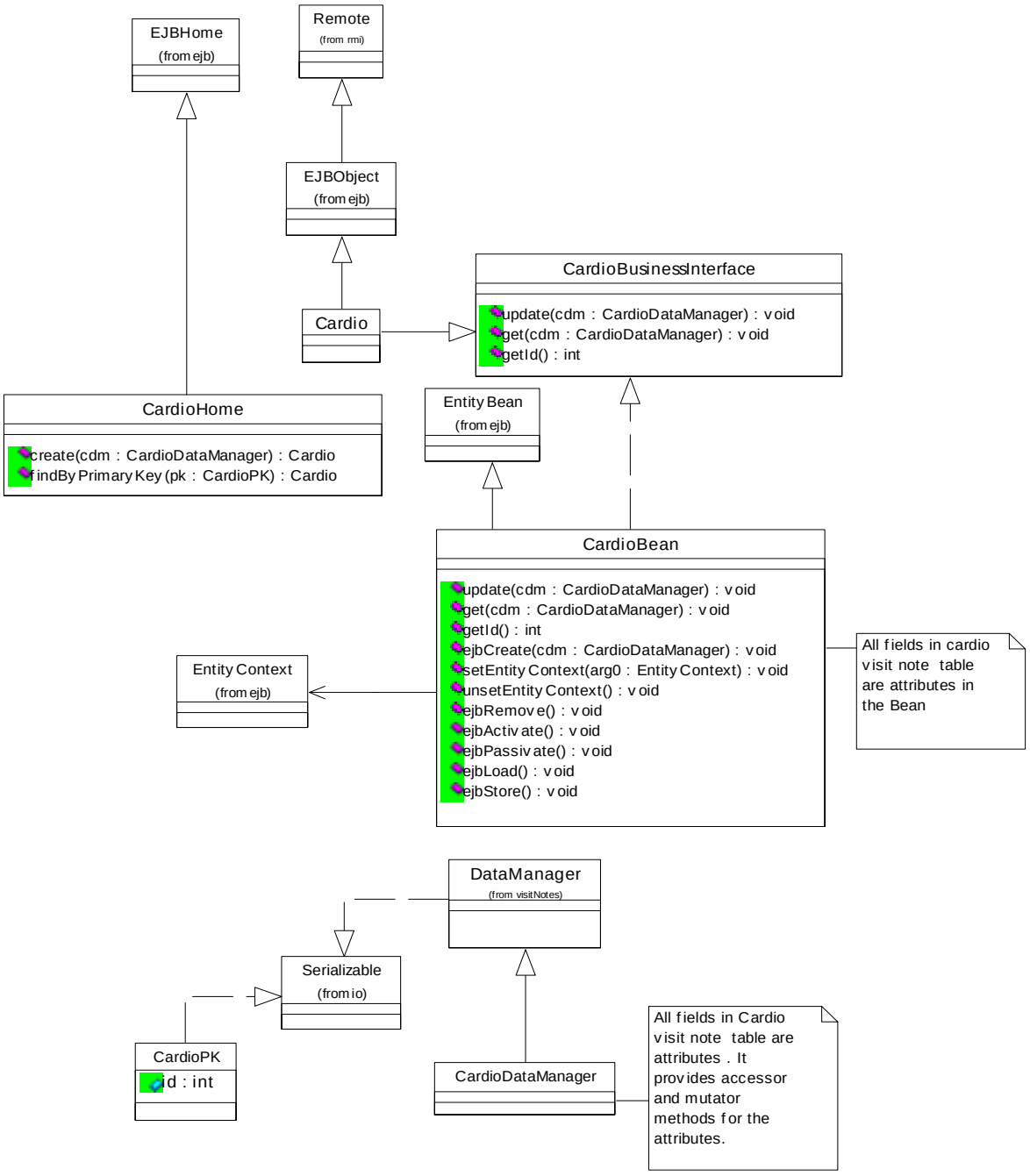

Fig 31. Design of Cardio Entity Bean

Fig 31 shows the design of an entity bean for the Cardiovascular category of the visit note. All the other categories employ a similar design. The entity bean also uses the business inteface pattern for the same reasons mentioned earlier. The primary key which is the patient id is mapped to the CardioPK object. The home interface provides a findByPrimaryKey method to locate records using the CardioPK object.

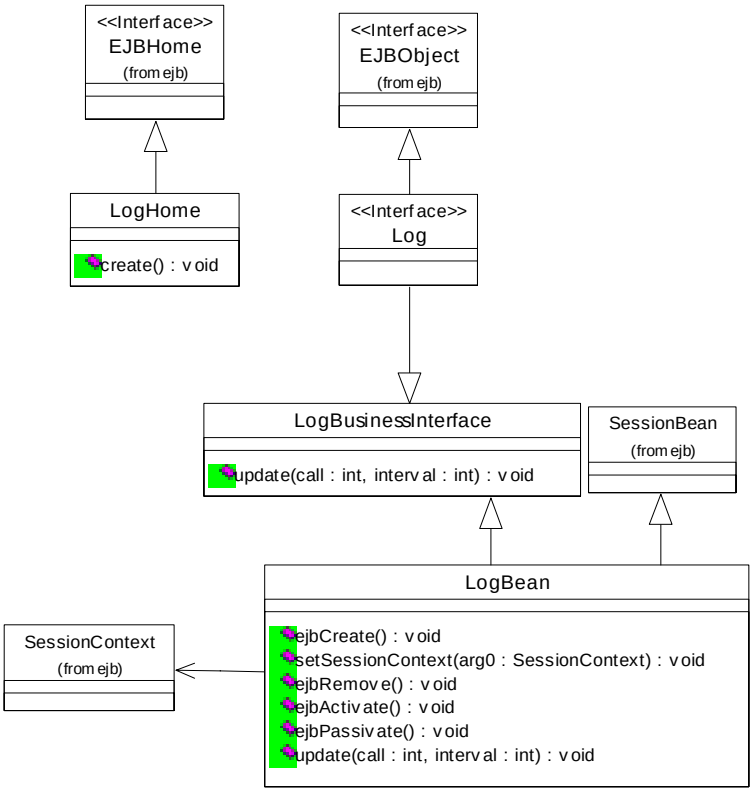

Fig 32. Design of LogBean

In order to analyze the performance of the EJB application it was necessary to log the time taken by each of the services provided by the VisitNotesBean. The time interval cannot be directly written to a file since the EJB specification prevents direct access to the file system from within the bean. Hence a Logging Server was designed which listens at a UDP server port and logs the

time stamps received into a Server log file. A LogBean which is a stateless session bean residing in the EJB server acts as a client to the Logging Server. The services of the LogBean are used by the VisitNotesBean to record the time stamp for each method.

The design of the LogBean is shown in Fig 32. It provides an update method which takes in the type of call, which could be a Get, Set or Update and the time interval. The update method creates a Datagram packet and sends it to the Log Server.

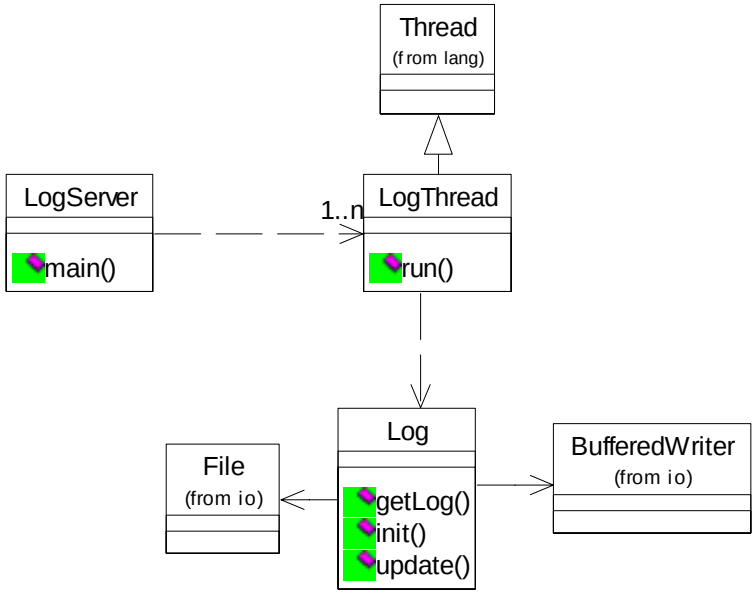

Fig 33. Design of Log Server

The LogServer spawns a LogThread each time it receives a packet. The LogThread uses the services of the Log class to log the packet data. It parses the data received in the packet, obtains a handle to the Log object and calls the update method to log the data received in the packet. The Log class implements the Singleton pattern [11]. The only way to obtain a reference to the Log object is through the getlog method. This ensures that the update methods when called by

multiple threads writes the log data to the same file. The init method is called to initialize the log file and record date, time of logging, title and other details. The update method is synchronized, as it may be called by multiple clients simultaneously.

In order to test the performance of the EJB system it was required to simulate multiple clients connecting to the server simultaneously. A multithreaded program was designed for this task. The client spawned a number of threads and each thread independently connected to the server to avail the services of the VisitNotesBean. The number of threads and the type of request to be made can be passed as command line arguments. The patient id for all requests was randomly generated. The type of request could be Get, Set, Update or Any. Get, Set and Update invoked the corresponding methods on the VisitNotesBean. If Any is provided as a command line argument, the threads were allowed to make any of the three calls randomly.

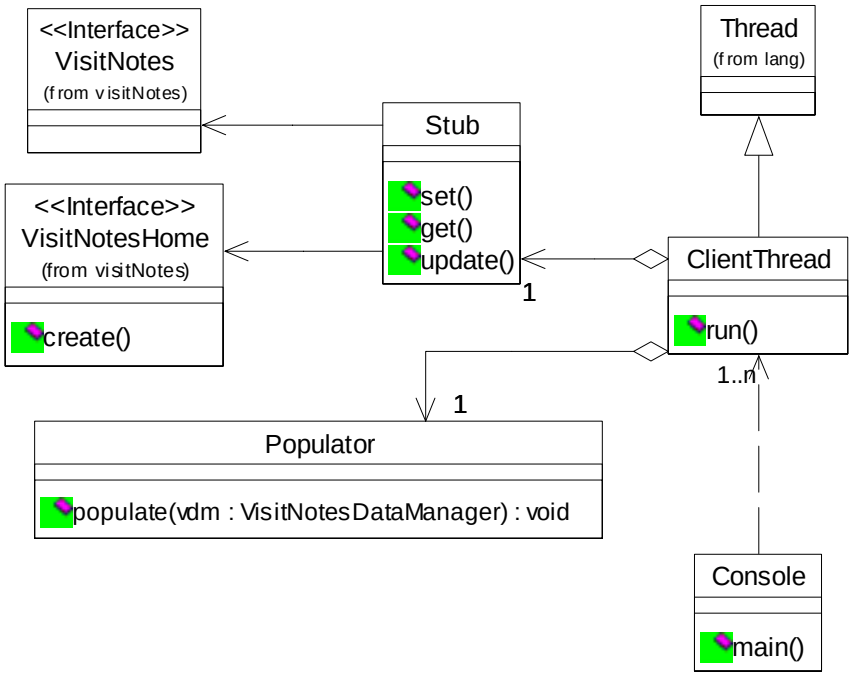

Fig 34. Design of Multi-client simulator

Fig 34 shows the design of the multi-client simulator. The Console creates the number of threads specified in the command line. Each thread aggregates a populator and a stub object. The populator is used to randomly populate the VisitNoteDataManager to be marshalled. The stub makes the actual calls on the the EJB server.

5 IMPLEMENTATION

This chapter discusses the various implementation issues, the tools used and the problems faced during implementation of the HHC System.

5.1 Implementation Decisions

The HHC System was implemented on Windows NT 4.0 platform using JDK 1.2, Weblogic 4.5.1 Application Server, Netscape Fast track web server, Gefion WAICoolRunner Servlet engine, Netscape Directory Server and Microsoft Certificate Server. The HHC System directly depends on the services offered by these commercial tools. The following sections discuss the criteria for choosing the above tools.

5.1.1 Choice of Implementation Language

The HHC System is a highly interactive tool. One of the main requirements was ease of use of the system by the nurse. Hence a extensive set of graphical user interfaces had to be designed for the nurse to interact with the HHC system. Another requirement was remote access through a standard web browser. Java language as defined in the Sun Microsystems Java Development Kit (JDK 1.2) was chosen as the implementation language since it provided a very easy and extensive development API for user interfaces and had direct support for web applications. Java enforces modular development thereby making future additions and modifications to the system simpler. Any application built in Java is portable across multiple platforms. In the event that the HHC client side application needs to be ported to a different platform, no additional effort would

be required as the entire application is written in Java. Also, the automatic garbage collection and memory management facilities in Java enable development of a robust application.

5.1.2 Choice of Development Environment.

The HHC application involved design of a large number of graphical user interfaces. Hence a Java Integrated Development Environment (IDE) which supported visual composition of graphical interfaces was a requirement. An IDE facilitates rapid application development. The IDE must also have a versioning system manage multiple revised versions of the software. The code generated by the IDE for GUI components must be easily comprehendable and modifiable. IBM VisualAge for Java Professional edition version 2 met these requirements and also provided automatic documentation features. Hence this tool was chosen to design the HHC client application.

5.1.3 Choice of Database

Apart from meeting the requirements mentioned in 4.2.1, the database had to be capable of storing image files. Although storing image files was not implemented in this version of the system, it is a feature which would be added to the HHC System in a future version. Also, since the implementation language was Java, the database must have Java Database Connectivity compliant (JDBC) drivers available. Oracle database provides capabilities to store file attachments as BLOBs and also had a range of JDBC compliant drivers available. Hence Oracle ver 8.0 was chosen as the database for the HHC system.

5.1.4 Choice of Web Server

The web server to be used for the HHC System had to meet three requirements

- Support SSL based security with client authentication
- Allow third party servlet engines to connect to the webserver as plugins
- Have built in support for Directory Server based access control.

Netscape Fast Track web server met these requirements and allowed third party plugin software to connect to the Web Server through a Web Application Interface (WAI) which is a CORBA based proprietary interface provided by Netscape. It also provided logging and online monitoring features. Hence the Netscape Fast Track ver 3.1 was chosen as the web server.

5.1.5 Choice of Directory Server

The directory server to be used had to comply with the Lightweight Directory Access Protocol (LDAP) and must be compatible with the Web Server. The LDAP was a requirement since the tree based hierarchial structure and flexiblity it provided for specifiying access control policies made it appropriate for the HHC System. The Netscape Directory Server met these requirements and was directly compatible with the Fast Track Web Serve. Hence Netscape Directory Server was used for the HHC System.

5.1.6 Choice of Certificate Server

The certificate server chosen must be capable of issuing client authentication certificates based on the SSL protocol with Microsoft Base Certificate Provider as the Cryptographic Service Provider since the HHC system runs on Windows NT and remote clients are assumed to be

running a web browser under Win NT. The certificate server should accept PKCS #10 certificate requests, verify the information in the request and issue certificates in the standard X509 format. The certificate server must also have a valid certificate authorising it as a certifying authority for the HHC System. The certificate server must also be capable of processing and issuing a server authentication certificate. This is required since the web server requires a certificate to authenticate itself to the client. Microsoft Certificate Server was chosen as it met all these requirements. The certificate server was authorised as a certifying authority by Verisign Inc, which is an accepted certifiying authority in the e-commerce industry.

5.1.7 Choice of Servlet Engine.

The design of the remote access system required a servlet engine which supported both Java Servlets ver 2.1 and Java Server Pages 1.0. Also, the servlet engine had to work with the Netscape Fast Track web server as a plugin through the Web Application Interface provided by Netscape. Gefion Software's WAICoolRunner is a servlet engine designed specifically for the Netscape Fast Track server and also provides support for JSP 1.0. Hence it was chosen as the servlet engine.

5.1.8 Choice of EJB Server

Although the EJB 1.0 specification did not mention entity beans as a mandatory requirement, the architecture of the HHC EJB System required an EJB Server which supported entity beans. The HHC System required the deployment of 15 EJBs. The process of development would be

much faster if the EJB Server supported hot deployment. BEA Weblogic Application Server 4.5.1 provided these features and hence was chosen as the EJB Server for the HHC System.

5.1.9 Choice of Database Connectivity Driver

The database connectivity driver to be used had to meet two requirements. It had to be compatible with Oracle ver 8.0 database and had to comply with the Sun JDBC 2.0 specification. Oracle Corp supplies two drivers which meet these requirements. A "Oracle thin driver" and "Oracle Call Interface [OCI] driver". The oracle thin driver is a platform dependent driver since it uses native calls to the database. The thin driver is a pure java driver and communicates with the database through java sockets using the Transmission Control Protocol (TCP). Since the HHC System had to support both remote clients as well as the HHC client application the Oracle thin driver was chosen as the database connectivity driver.

5.1.10 Choice of Video Conferencing System

The video conferencing system had to provide for still image capture and real time audio-video conversation. It had to work over regular telephone lines and the cost had to reasonable for the HHC system to be feasible. A Via TV VC1050 set top box, a Howard NCK41CV telecamera and a 13-inch Orion color television set formed the video conferencing system at the patient site. The nurse computer was equipped with a Aver Media TV tuner card, a set top box and a telecamera similar to the one at the patient's home. This combination met the requirements and the total cost was below $1000 per set.

5.2 Implementation of Nurse televisit station

The nurse televisit station consists of the video conferencing system and the HHC Client Application described in section 3.2. The video conferencing system was kept independent of the client application. The TV tuner card provided an interface to freeze images for capture, control zoom, etc. In the current version, captured still images are stored in the local system. The televisit station requires the Java Runtime Environment (JRE 1.2) in order to execute the application.

5.3 Implementation of Administrator and Data Entry Operator Systems

As described in section 4.6, the same client application is used by the nurse, data entry operator and the administrator However the administrator and the data entry operator terminals need not have the video conferencing system as this facility is used only by the nurse. All other implementation considerations remain the same for all types of users.

5.4 Implementation of Remote Access System

The IBM VisualAge IDE does not provide support for Java servlets. Hence the development of the remote access system was done using the Sun Java Server Web Development Kit (JSWDK). This kit supports the Java Servlet 2.1 and the JSP 1.0 API. The HTML content of the JSP pages was developed using Netscape Composer and the JSP tags were embedded into the HTML file. The directory server and the web server resided on the same machine. Both the directory server and the web server provide a local and a web based administrator console. For security reasons the access to the administrator console was restricted to the local machine.

5.5 Implementation of the HHC EJB System

Clients can connect to a EJB server using either the Internet Inter Orb Protocol (IIOP) or through Java Naming and Directory Interface (JNDI). IIOP provides language and platform transparency and clients can be written in any programming language in any platform. JNDI provides naming and directory functionality to applications written in the Java programming language. As the client was implemented in java, JNDI was used since it provided a much simpler interface. In order to use JNDI one of the following service providers must be available

- Lightweight Directory Access Protocol (LDAP)
- CORBA services (COS) naming service
- Java Remote Method Invocation (RMI) Registry

BEA provides an implementation of the JNDI specification as a part of the WebLogic Application Server which uses Java RMI as a service provider. This implementation of JNDI was used to connect the HHC EJB client to the server.

5.6 Problems Faced

Since the video conferencing system and the client application were kept independent of each other, it was not possible to map the still images captured during the televisit directly into the database. This would have been possible if the two systems were integrated but no commercially available TV tuner card provided an API for integration with other software systems.

Image capturing gave best results at high resolution and when the wound was held very close to the camera. However, obtaining focus and having the patient remain still at the point of focus for capture of images was a cumbersome process.

Netscape Navigator and Internet Explorer, the two standard web browsers did not render the JSP pages uniformly. There were variations in font and positioning. The solution was to implement separate pages for each type of browser. However for this prototype the JSP pages were written conforming to Netscape Navigator.

For analysing the performance of the EJB server, multiple client connections were simulated using threads. This is not exactly the same as multiple clients connecting from different machines as all the client threads connect from the same machine. However from the server perspective, each thread makes a separate connection and obtains a handle to a separate instance of the remote interface of the session bean. This helped analyse server side performance, but true client side latency could not be determined.

6 ANALYSIS AND CONCLUSION

6.1 Analysis of HHC EJB System

The multiclient simulator discussed in 4.5.1 was used to simulate multiple clients connecting to the EJB server. The Log Server logged the server side latency i.e, the time taken by the server to complete a request. The latency was logged for all three types of requests ie, Get, Set and Update.

As discussed in section 4.5.1, for the Set request a VisitNotesDataManager object is marshalled to the server and the VisitNotesBean which is a stateless session bean fulfills the request by communicating with a set of entity beans. The server side latency for the Set request when multiple clients connect is shown below

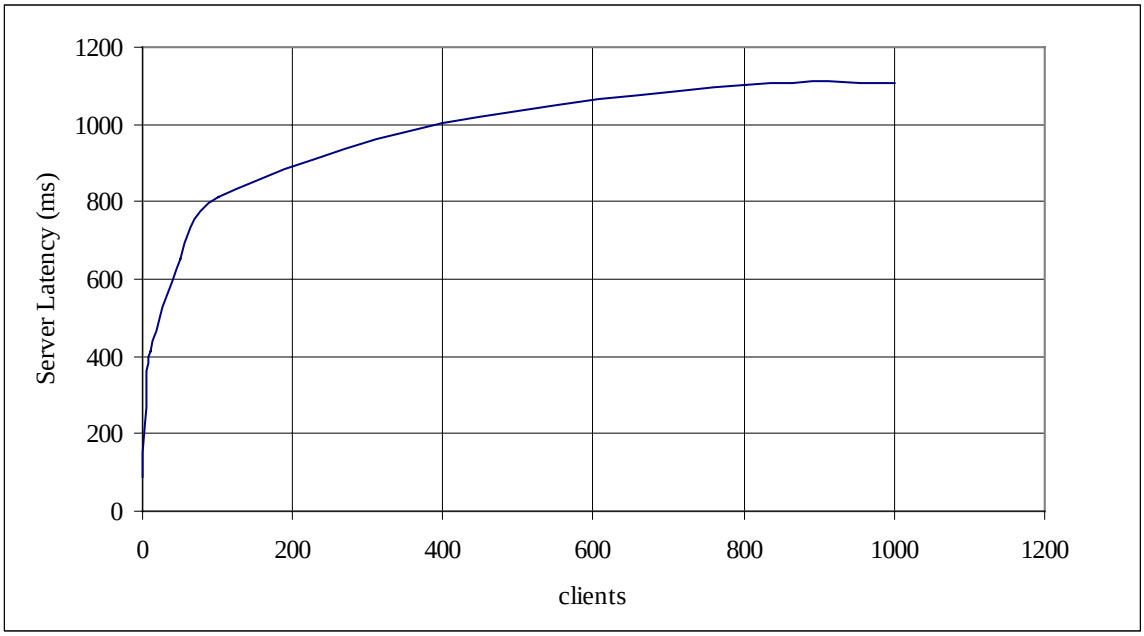

Fig 35. Latency for Set request

Clients	Set(ms)
1	90
5	314
10	414.6
50	654
100	810
400	1002
800	1100
1000	1105

Table 1. Latency for Set Request

As can be seen, the latency increases as the number of clients connecting at a given time increase and finally stabilises around 800 connections with a response time of about 1100 milliseconds. When the number of connections go beyond 800, the server latency remains fairly constant, this is expected because the number of concurrent requests the server can handle at a given time is bounded by the speed of the machine, memory, number of connections in database pool, number of beans in free pool etc. When the server reaches this limit the remaining clients wait until some requests are completed. Fig 35 only shows only the time taken for the request to be handled by the server once processing of the request begins. The waiting time is not recorded. However when the number of clients increases, the waiting time for each client increases proportionately. Given the EJB architecture, with the container managing the handling of all requests it is not possible to determine the waiting period for each request from within a bean. The client side latency also cannot be truly determined from the multithreaded client simulator used in this thesis. In order to determine the true client side latency it would require multiple clients connecting from separate machines or atleast separate address spaces at the same time.

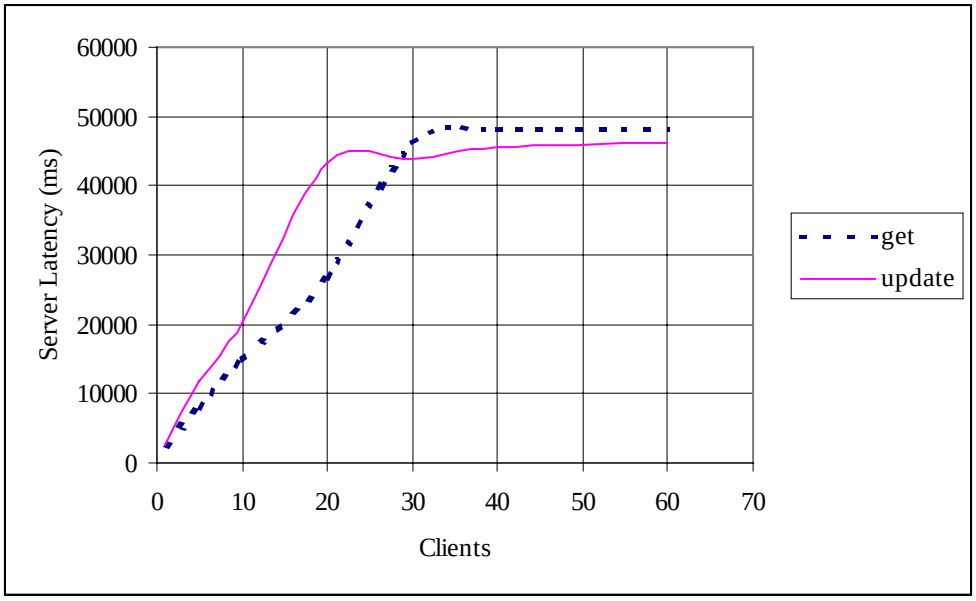

Fig 36. Latency for Get and Update Requests

Clients	Get (ms)	Update(ms)
1	2634	2614
5	8200	11892
10	15041	20570
20	26949	43200
30	46051	43900
40	48056	45500
60	48041	46100

Table 2. Latency for Get and Update Requests

Fig 36 shows the latency for Get and Update requests. It can be observed that these requests are atleast 25 times slower than the Set request. The reason for this is as follows.

The Set request creates new entity bean objects by calling the ejbCreate method on each of the entity beans. This does not require any database access as the Set method makes only an in-memory creation and persistence of the new entity bean is done by the container at an appropriate time after the completion of the Set request.

Whereas a Get request must perform the following

1. Locate entity bean objects by calling the findByPrimaryKey method
2. Copy the state of each of the located entity beans to the references in the VisitNotesDataManager façade.

The first step involves database access in order to obtain entity beans which map to the data requested. Database access takes a considerable amount of time. Also, it must be noted that the VisitNotesDataManager façade references over 10 visit note categories and each category contains 10 fields or more. Hence the second step is also time consuming.

The update method must perform the following

1. Locate entity bean objects by calling the findByPrimaryKey method
2. Modify the state of all these entity beans to those specified by the client

Step 1 is same as the one performed by Get. Step 2 involves updating every field of every entity bean object. Hence, this step is also time consuming.

For these reasons, the Get and Update methods have a much higher latency as compared to the Set method.

In Fig 36, it is observed that the update method stabilizes with fewer clients and provides a slightly better performance as compared to the get method. This is because modifying the state of the located entity beans takes less time than copying the state of the entity beans into the object marshalled by the client.

It must also be noted that while the Set and Update methods have serializable isolation, ie, the transaction management property of set and update methods is set to TX_SERIALIZABLE, the

isolation level of the Get method is set to TRANSACTION_READ_UNCOMMITTED. Get requests do not modify the database, and although setting the isolation level to read-uncommitted allows dirty, repeatable and phantom reads, this is not likely to happen very often in a real world situation in home care. That is to say, if the nurse is updating the visit note while a physician or any other remote client tries to read the visit data, it might lead to dirty, repeatable or phantom reads. This however is not likely to happen often, and if it does ever happen it does not lead to any major problem as the physician can always verify the information once the visit is complete. In the unlikely event that the physician looks at an ongoing visit, the server may perform dirty reads and return a partial visit note (which is a snap shot of the current state of the entity beans) to the physician.

6.2 Analysis of Clinical aspects

The prototype described in this thesis was deployed by the Monongalia County Health Department [MCHD], linking 2 patients at their residence with the home care department of MCHD. Patient monitoring devices and the video conferencing system were installed at the homes of the patients. The patients chosen were about seventy years old and were suffering from Type II Diabetes. The software provided a clinical pathway for Diabetes. About 10 electronic home visits were conducted and data collected on the technical and clinical aspects of the project.

The total cost of the set up at the patient's residence is around $800.00. The nurse station including the cost of the computer would be around $2750.00.

The average time taken for a televisit was approximately 35 minutes. This included capturing vital signs, discussing medications and assessing other problem areas. The vital signs recorded were temperature, blood pressure and pulse. The patients were requested to read out the instrument displays. Capturing vital signs took about 15 minutes.

Image capturing gave best results at high resolution and when the wound was held very close to the camera. Obtaining focus and having the patient remain still at the point of focus for capture of images was a cumbersome process. The images captured helped the nurses monitor progress of a wound over a period of time. However the quality of still images was not good enough to be used for any diagnostic purposes. About one third of the physical visits could be converted to televisits through this system, in our opinion.

Limitations

The system does not provide any facility to transmit the vital sign readings from the patient residence. The nurse has to rely on the patient, for measuring and reading out the instrument displays. Hence patients who have limited eye sight or those who cannot operate vital signs instruments on account of physical limitations cannot be treated through this system.

Needless to say, nursing visits which undertake any intrusive procedures, such as dressing a wound or administering an injection, cannot be conducted by this remote televisit system.

6.3 Summary and Conclusions

This thesis presented a low cost approach to secure telemedicine for home health care. The objectives of the system as stated in the first chapter have been met successfully. The system is user friendly and provides a simple interface for the nurse.

Although the analysis of the EJB server is done specific for the home care application, the conclusions drawn may be applied to any EJB based system. In interacting with a database, all applications typically perform four operations SET, UPDATE, GET or DELETE. The performance of the EJB server when multiple clients try to SET, UPDATE or GET large amount of data through a façade which references a set of fine grained objects is analysed and described in section 6.1

Commercially available home care telemedicine systems and other research projects described in section 2.1 mainly focus on high resolution, high frame rate video conferencing systems. Although the approach presented in this thesis does not provide a very high resolution video conference facility, it does address other issues of great importance, namely: cost, security, confidentiality of patient data, remote access from anywhere and scalability. Moreover it is designed as an extensible system, built on accepted modern standards and open systems.

A physician can monitor patient progress through a standard web browser. In case of a medical emergency authorized paramedics only need a computer with Internet access to obtain the patient's past medical history.

The EJB version of the system can be used to provide a central database for all home care agencies in the state or country. If patients relocate, continuing care can be provided since all data is accessible through a central server.

The HHC System described in this thesis needs improvement in the following areas

- As can be seen in Fig 36, the server latency can be as high as 40-50 seconds if around 30 clients connect at the same time. This is mainly due to the quantum of data being transferred between client and server. Hence in order to obtain a reasonable performance in case the

number of clients exceed 20, it will be necessary to distribute the EJB's across multiple application servers through a clustering mechanism and use higher power servers.

- The video conferencing system is independent of the HHC software system. Hence it is not possible to directly capture still images directly into the database. If a TV tuner card which provides a driver with an API for integration is used, this limitation could be overcome.
- Although the façade pattern used on the server side provides a clean interface and reduces object granularity and network traffic, the design does not easily lend itself to future enhancements. Addition and removal of new visit note categories would require modification of the Get, Set and Update methods in the session bean façade.
- Vital signs data cannot be transmitted from the sensors at the patient's residence through the telephone line. The nurse relies on the patient for measuring and reading out displays. The system can be enhanced to provide this facility by using network routers which integrates the video conference and the vital sign data and developing the necessary software and protocols for the same.

In its current form, the system provides a feasible low-cost approach to providing telemedical care to home based patients.

BIBLIOGRAPHY

1. Eisler, P. (Aug. 17, 1997). “Fevered growth could turn to consolidation.” USA Today**.**
2. Basic Statistics About Home Care 1996. (1996).
3. Peterson, C. (1997) “Home health care to continue to grow.” Managed Healthcare**,** 7(4), 28-30.
4. Telemedicine Today "Home health Care via Telemedicine" Fall 1995 26-27
5. Tele-Home Health: Kaiser Permanente Medical Center’s Pilot Project, July 1997
6. Pushkin, D. S., et al. (1995). Letter to the editor: joint federal initiate for creating a telemedicine evaluative framework. *Telemedicine Journal*, 1(4), 395-399.
7. M.F Burrow, M. E. Stachura, K. Grisby, L Schlachta. The Electronic House Call: A Telemedical approach for delivering health care to patients in their homes. American Telemedicine Association April 1997 Conference
8. Ace Allen. Managing Home Nursing visits electronically: A quantitative analysis.
9. Takano T, Nakamura K, Akao C. Assessment of the value of videophones in home healthcare. *Telecommunications Policy*, 19,3:241-248, 1995.
10. K Mahumud, Telemedicine to the Home: The American Telecare approach. *Telemedicine Journal*, 1(2), 1995
11. E Gamma, R Helm, Ralph Jhonson, John Vlissides: Design Patterns Elements of Reusable Object-Oriented Software. Addison Wesley 1995
12. S Rao, M Xing Design Java servlets with the Delegation Event Model *JavaWorld*, Sept 1999
13. J Zukowski, M Loukides: Java AWT Reference, O'Rielly & Associates April 1997
14. J Gallimore, Implement Enterprise JavaBeans JavaPro 1999
15. Elizabeth, Dorothy and Denning, Robling. *Cryptrography and Data Security.* Addison Wesley, Reading. MA January 1983
16. R Monson-Haefel. Enterprise Java Beans O'Reilly & Associates June 1999
17. T Valesky. Enterprise Java Beans Addison Wesley, Reading MA August 1999
18. D.R.Nilsson, P.M.Jakob, B. Sarantakos, R A Stinehour. Developing JavaBeans using VisualAge for Java, Version 2, Wiley Computer Publishing 1999

19. J Hunter, W Crawford. Java Servlet Programming O'Reilly & Associates March 1999
20. D Flanagan, Java in a Nutshell. O'Reilly & Associates May 1997
21. D. R. Callaway. Inside Servlets, Server-Side Programming for the Java Platform. Addison Wesley, Reading MA 1999
22. J Engel. Programming for the Java Virtual Machine. Addison Wesley, Reading MA 1999
23. D Lea. Concurrent Programming in Java. Design Principles and Patterns. Addison Wesley, Reading MA 1999
24. BEA Weblogic Inc. Weblogic API Reference Manual 4.5 BEA Systems 1999
25. BEA Weblogic Inc. Weblogic Developers Guide BEA Systems 1999
26. Sun Microsystems Inc. Enterprise JavaBeans Specification 1.1, Sun Microsystems 1999
27. A Thomas. Enterprise JavaBeans Technology Server Component Model for the Java Platform Patricia Seybold Group Dec 1998
28. E Roman, R Oberg The Technical Benefits of EJB and J2EE Technologies over COM+ and Windows DNA. The Middleware Company, Dec 1999
29. Sun Microsystems Inc. Java Servlets API Specification Version 2.2, Sun Microsystems 1999
30. Sun Microsystems Inc. Java Server Pages Specification Version 1.1, Sun Microsystems 1999

APPENDIX I -Specification Of Video Conferencing System

Via TV 8x8 VC 1050 phone

SYSTEM [ITU - T Standard H.324]

Signaling	H.245
Channel Multiplexer	H.223

VIDEO [ITU-T Standard H.263]

Resolution	
CIF	352 x 288 pixels
QCIF	176 x 144 pixels
SQCIF	128 x 96 pixels
Frame rate, max(SQCIF)	15fps
Video Output	
Main monitor	Composite
Format	NTSC

NETWORK INTERFACE

POTS (Plain Old Telephone Service)	RJ-11
Transmission speed (max)	33.6 Kbps (V.34.12)
Communication mode	Synchronous(V.80)

Telecamera Howard NCK41CV

Scanning System	NTSC or S-Video
CCD Imager (Pixels)	768(H) x 494 (V)
Horizontal TV Lines	450
Sync System	InterSync
Electronic Auto Iris	YES
Minimum Illumination	8 Lux
Varifocal f3.5mm-f8.0mm, F1.8Lens	YES
Focal Distance	300mm - ∞
Video Output	Composite / Component

TV Tuner Card - Aver Media TV-Phone

Input Signal	75 Ohm Coaxial TV Antenna Input
	S-Video Input / Composite (RCA Video Input)
Output Signal:	Audio Output (Direct to Speaker or Loop to Sound Card)
Image/Video Capture	Full Size Capture Capability (Upto 640 x 480 for NTSC