DESIGN ISSUES IN THE KNOWLEDGE BASED E-HEALTH PROJECT

by

Pai-Fang Hsiao

Honour B. A. with High Distinction and M.A.
University of Toronto, Canada

Submitted to the Department of Civil and Environmental Engineering
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Civil and Environmental Engineering

at the

Massachusetts Institute of Technology

June 2002

Signature of Author ______________________________
Department of Civil and Environmental Engineering
April 23, 2002

Certified by ______________________________ [illegible]
John R. Williams
Associate Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by ______________________________ [illegible]
Oral Buyukozturk
Chairman, Department Committee on Graduate Students

DESIGN ISSUES IN THE KNOWLEDGE BASED E-HEALTH PROJECT

by

Pai-Fang Hsiao

Submitted to the Department of Civil and Environmental Engineering on April 23, 2002 in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Civil and Environmental Engineering

ABSTRACT

In this thesis, I outline some of the design-related issues that came up during the design phrase of the Knowledge Based E-Health Project.

The goal of the Knowledge Based E-Health Project is to improve the quality of health care by providing self-educational resources to patients, allowing them easy access to their own medical records and focusing on preventive care with self-monitoring tools. It also facilitates information flow for the health care professionals, making data entry, update and maintenance easy and allowing easy access to data from virtually anywhere and through various channels, including wireless devices.

I discuss factors that influenced our final design and implementation decisions. Such factors include time constraints, choices of technology, compatibility issues among different technologies, resource constraints, complexity of debugging and testing, security issues, database ownership issues, legacy database issues, legal issues, etc. I compare the original architecture and the actual implementation and discuss the differences as a result of these decisions. Finally, I evaluate our project in the context of real-life industry projects and comment on what additional features we need to implement in order to make our product marketable.

Thesis Supervisor: John R. Williams
Title: Associate Professor of Civil and Environmental Engineering

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

Many people have made this MEng year a rewarding, enjoyable and unforgettable experience. I have been looking forward to thanking each one of them

I thank Li-Wei Lehmen, Judson Harward and Steve Lerman at Course 1 for encouraging me to apply to the MEng program. To a certain degree, they are responsible for me being here ☺ (it seems appropriate for an IT thesis to contain a smiley). I also thank Diane Massam and William Green for their support and help.

I would like to thank John R. Williams, our thesis and project supervisor, for his guidance and support and for introducing us to cutting-edge technologies. I am indebted to my academic advisor at Course 1, Eric Adams, for being so encouraging, supportive and willing to go out his way to help me in any way he can. I am also grateful to Cynthia Stewart and James Riefstahl for their help with administrative matters.

I thank my team members: Sakda Chaiworawitkul, Andrew Ferriere, Glorimar Ripoll, Osamu Uehara and Tashan Yen for the learning experience, team work, fun and moral support. We have complemented each other's work throughout the year. It has been fun working with them. More importantly, we have become good friends, who are there for each other in times of need. Thank you guys.

I also would like to express my deepest gratitude to James Goodwin, Emily Farmer and Hiro Miyamoto at IBM Software Group in Cambridge, Judson Harward at CECI and George Kocur for their helpful discussions during the time when I was designing the architecture for our project.

My colleagues and mentors at IBM Software Group in Cambridge have been a tremendous support. They always help me keep things in perspective. I thank my manager, Jim Flynn, for being one of the most supportive managers I will probably ever have and for being a friend. I thank my mentor Emily Farmer, for her words of encouragement, for caring and for being there. I thank Hiro Miyamoto, who has taught me how to do my job well, who has never failed to deliver and has always managed to assure me that everything will be ok when I panicked. I thank Lucille Parker, Robin Rouse, Dan O'Connor, Siobhan Donnelly, Simon O'Doherty, Charles Pau, Scott Brumit, Pete Myers, Yuko Haraki, Christina Martarano, Myong Ho Choi, Otomo-san and Anny Wang for the support, help and fun at work. It has been an honor working at IBM.

Finally I would like to thank my family and close friends, whose support and love have kept me going throughout the years. Even though I have chosen to leave out private details, their presence in my life undoubtedly matters the most. My gratitude and love go to my parents, Shawn, Jason, Sheng, Bora, Meltem, Barry, Alex, Angela, Rafael, Chris, Philipp, Christiane, Pati, Diane, Elan, Keren, Milan, Caralee, Gerd, Anita, Matthias, Fei-Wen, Sharon, Roger, Julia, Tse-Hsuan, Lina, Tatjana, Armand, Steffi, Stephan, Jingyi, and Jung-A. I especially want to thank Meltem, Bora, Sheng, Alex, Angela, Rafael, Chris and Philipp for keeping me sane, for always having faith in me, for listening and for providing a shelter when I needed to escape. *Schön, daß es Euch gibt.*

GLOSSARY OF ACRONYMS

ADO. Microsoft ActiveX Data Object

ASP. Active Server Pages

HMO. Health Maintenance Organization

HTTP. HyperText Transport Protocol

IDE. Integrated Development Environment

IMAP. Internet Messaging Access Protocol

LDAP. Lightweight Directory Access Protocol

ODBMS. Object Database Management System

ODBC. Open Database Connectivity

OLE. Object Link Embedding (note Microsoft no longer ascribes this meaning to the word OLE)

ORDBMS. Object Relational Database Management System

PDA. Personal Digital Assistant

RDBMS. Relational Database Management System

SQL. Structured Query Language

SSL. Secure Socket Layer

TCP/IP. Transmission Control Protocol/Internet Protocol

XML. Extensible Mark-Up Language

1 Introduction to the Knowledge Based E-Health Project

The fast growing health care industry is witnessing some significant trends that need to be addressed to retain industry stability. Particularly, control of escalating costs and quality of patient care are keys to any effective strategy. These issues, together with the recent introduction of information technologies in the health care system and the rising wave of patient consumerism, are pushing the industry to adopt a new approach to health care. For any new approach to be successful, it must prevent the health care industry from reaching the unsustainable state that many experts predict for the near future.

Many traditional and modern disease-management companies have been successful at reducing costs and improving quality of care. However, leaders of the health care consulting practice expressed "no company has yet assembled anything that resembles a full platform of capabilities" (Matheson and Robinson 2000). The approach we envision will fill the existing gap in current market offerings.

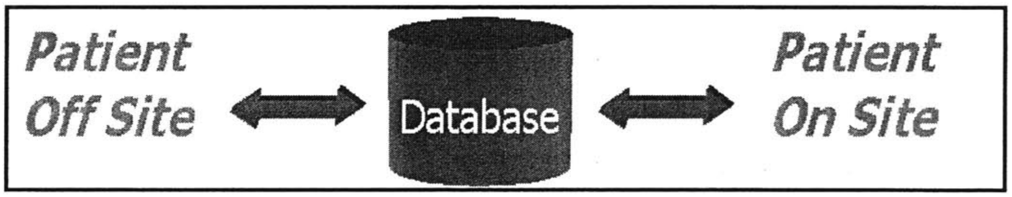
The three elements of our approach are: evidence-based treatment, outcome-focused treatment, and utilization of best practices. Furthermore, to truly differentiate our product, information technologies are at the heart of our strategy. Using emerging technologies, such as Microsoft's .NET Platform, we intend to transform hospitals into knowledge centers. By effectively and seamlessly integrating on-site and off-site knowledge, hospitals will witness benefits of cost reduction, higher patient satisfaction and better health care quality.

The Knowledge-Based E-Health system consists of (1) a web application/client with user-customized views, (2) a web service to facilitate database interactions, and (3) tools to manage and control access to the databases.

1.1 Evidence-Based Treatment

This element emphasizes ongoing health maintenance and feedback from health care professionals. Traditionally, patient's information is collected only when the patient is on-site, i.e., at the hospital. Likewise, feedback is only received on-site. Evidence-based treatment allows the patient to transmit and receive information to and from the physician regularly (even when the patient is off-site). Figure 1 illustrates the proposed flow of information to and from the medical database. We believe that prevention and proper monitoring of patient conditions will reduce costly hospital visits and provide higher quality of health care.

Figure 1. Off-Site and On-Site Database Interaction



1.2 Outcome-Focused Treatment

Traditionally, physicians work on optimizing a particular individual procedure or event. They deal with medical events as if they were unrelated to other events, because they want to focus on solving the problem at hand. We believe that through a systematic approach, physicians will focus on optimizing the overall outcome of the patient, not just individual sessions.

1.3 Best Practice Guidelines

To develop best practice guidelines, it is necessary to analyze the utilization of resources in a given situation over a number of episodes. The database needs to be effectively analyzed to find the most effective practice in any given situation. Physician may choose to deviate from the guidelines based on professional judgment. In any case, we believe these guidelines promote a consistent level in care delivery.

1.4 Approach Benefits

By implementing these three elements, a control can be exercised on the number of serious cases that arrive at the hospital. A system that monitors patients off-site will help prevent, diagnose, and treat conditions to avoid reaching the critical level that becomes so costly for health care providers. For the patients, it is clear that they will benefit from constant attention from doctors. Physicians will be able to reduce the high workload associated with severe cases, allowing them to maintain high and constant quality of care. Furthermore, on a macro level, for hospital operators and other industry players, the approach will promote best practices to control quality of care and escalating costs.

2 Technical Issues

Technical issues that were considered include: (1) the technical implementation of the three elements of the approach, (2) medical data security, and (3) database ownership and legal issues.

These technical issues are self-explanatory, though not straightforward; we need to transform an idea into a new technological solution. This includes system design, architecture, and implementation, among other things. Also, technical decisions are strongly linked to the business environment. Thus, it is essential to consider them in the context of relevant business issues.

2.1 Technology Implementation

We are using Microsoft .NET, which is an emerging Integrated Development Environment (IDE), as the development platform. The predominant programming language we are using in generating our code is C#, an object oriented language designed to take advantage of features available to both Java and C++. Java Script is also used to animate the web pages, through which users interact with our system. Regarding database management systems, we have originally adopted an object relational model. The specific vendor we have chosen is Oracle 8.1.7. However, in the actual implementation, we opted to go with Microsoft SQL Server instead and in the end adopted a predominantly relational database model rather than object relational database model (refer to Section 4 below for further details about the reasoning behind these changes). For data security such as password encryption, we are employing the built-in encryption provided by .NET.

Evidence Based

Our solution focuses on ongoing health maintenance with timely feedback from health care professionals. Through our web service, patients can input values their physicians advise them to monitor on a regular basis. The web service is implemented as a .NET web service project for the back end and web application for the front end/client application. These inputs will be inserted into a table named 'SelfMonitor' (note that some of the table names in the final implementation of the database have been changed. The naming convention in this thesis, however, is consistent with the class diagrams and database modeling given in Appendix B and Appendix C for clarity and consistency). We will implement a flag mechanism in C# to check against patients' base values. If the inputted values are of concern, the patients' physicians will be contacted, both via e-mail and as an alert message when the physicians log on to the system. The physicians will then be able to contact the patients for additional information, advise them to come in for a visit, or give them consultation as to what the patients should be doing. For actual implementation details, see Section 4 below.

This service was the main focus of our first milestone. We have also included a scheduling service for the patients to view their scheduled appointments and to make new appointments according to their physicians' availability. They can also request an update on their profile information such as address change, marital status change, etc. For the physicians, they receive alert messages from the flag system when they log on so that patients who require immediate attention are taken care of in a timely manner. Our service also provides physicians with an overview of their daily appointments/obligations to help them stay on top of things.

Systematic Approach – Outcome Focused Treatment

The systematic approach element emphasizes the overall outcome optimization, instead of the optimization of individual procedures or components.

This is achieved through data mining and fast database query. Health care professionals will now be able to get a better overview of the resources (e.g. procedures, medication, etc) being used for a given case. The input will come from separate health care departments so that the effectiveness of the combined resources can be evaluated more efficiently. We have chosen object relational database management system to allow faster query to the database and at the same time allow storage of complex patient data such as pictures, graphics, audio files, etc.

This service was our target second milestone. However, in Section 4 Final Implementation below, I discuss certain resource and legal constraints we have in attaining sufficient data from various sources, which makes data mining hard to implement in our system at this stage.

Utilization of best practice guidelines

This element aims to produce an efficient analysis of the database to find the most effective practice in any given situation. In order to provide this data-mining service, we need to allow fast database query; and, since the query could be a combination of multi-purpose, we need to provide easy joins of tables. This is one compelling reason in favor of adopting relational and object-relational databases (see more discussion below). In the current design (see the section below on Database Modeling), each hospital will have a view "Statistical Data" for this purpose. Since each "Statistical Data" view has the same number and order of columns, it is straightforward for records from different hospitals to be accessed at the same time. We simply need to insert all the records into one table for easy access. To simplify the design, we have not addressed the questions of how legacy database systems are to be integrated and how data from such systems are to be transferred into the uniformed format of the "Statistical Data" view.

This service was our target third milestone.

2.2 Medical Data Security and Privacy

Security and privacy are the most important considerations when dealing with medical data, as they are not only protected by law but also our top commitments to our customers. We ensure privacy by enforcing strict access control to patients' data. For example, the statistical data contains no personal information; an HMO has no access to patients' self-monitoring data, etc. Similarly, external referral physicians are only allowed access to dada from patients referred to them for care. Appendix A lists access rights, in SQL commands terms, different groups of users have (see below for more discussions on the Role-based model).

With respect to data security, since ASP.NET does not provide data encryption (Platt 2000), in the original design, we planned to encrypt data by using the Secure Socket Layer 3.0, which is the dominant protocol for browser-server communications. In addition, we employ the built-in password encryption functionality provided by .NET to store users' passwords in the database.

2.3 Database Ownership and Legal Issues

The database ownership model we select will directly affect our business model. For example, it may have serious implications on what our revenue source will be. It may also affect the interactions between the system and the end-users. Furthermore, at a technical level, the ownership model will affect the system architecture and design, and the security and data accessibility features we need to implement. We considered three database ownership models:

- Localized (every hospital maintains its own database)
- Centralized (a central unbiased organization, e.g. the government, controls and owns patients' data)
- Third Party (a third party, such as an insurance company, controls and owns the data)

We explored each model's implications and studied how hospitals and potential competitors are dealing with ownership and security issues in North America. Based on our findings, we have chosen to implement the Localized model. According to this approach, each hospital maintains its own database and authorized local IS administrators control access to different parts of the database. There are several advantages to this approach.

First of all, changes made to the database are immediate. Under the Centralized or Third Party model, individual hospitals might not have enough access rights to modify records in the central database directly. They might need to submit an update request first and wait for the changes to be implemented.

Secondly, having a localized database makes it easier to keep access control list up-to-date. For example, if the ER is short-staffed and paramedics need to temporarily perform administrative tasks such as pulling out patients' hospital visit records, the local IS administrators can temporarily change the role of certain paramedics to the role of administrators to reflect current needs of the hospital (see Section 3 System Architecture and Design for a more detailed discussion on role-based security we are implementing).

Thirdly, having a localized database also makes individual hospitals less dependent. Under the Centralized or Third Party approach, if the server hosting the database is down, no single hospital will be able to access its records. Under the Localized model, on the other hand, data can be accessed smoothly even if other hospitals' servers are down. In addition, since database is managed by local IS administrators, they can attend to problems immediately.

In the case where a patient changes his/her health care plan and moves to another hospital, the patient will still have access to his/her medical records, primarily because of the web-based features we will implement.

However, as will be discussed in Section 4 Final Implementation, the reasons for adopting a Localized model are mainly logical and intuitive, i.e. they make sense and seem to facilitate the work of health care professionals in an optimal way. Having said that, it is necessary to conduct a reality check on the legal status of these issues, which might not always be intuitive.

3 System Architecture and Design

This section outlines the original architecture and design we came up with in January 2002. As discussed in the previous section, we have chosen to adopt the Localized model in terms of database ownership. Our architecture and design thus reflect this decision. We also incorporated as many features and use case scenarios as feasible in the original design, which represents the ideal feature specifications. However, as is often the case in the software industry, one needs to constantly reprioritize and modify the feature sets to meet deadlines or/and budget limitations. These issues will be discussed in Section 4 below.

3.1 Basic Architecture

The original architecture for our solution is illustrated in Figure 2 (the actual implementation and the differences between the original and final architectures will be discussed in detail in Section 4 Final Implementation). Under the Localized model, each hospital maintains its own database. We also take into consideration that some hospitals may already employ certain legacy database systems to manage basic data like patients' profiles. We would like to incorporate these data into our system as well. In addition, we also allow storage of complex data format such as images, video, signature files, etc. in the database design, i.e. by employing an object-relational database model, which also allows faster query time.

For each hospital, we will build a web service using .Net to access the databases. The .Net platform provides built-in classes (i.e. Connection, DataAdapter, Command, and DataReader) to support data access in MS SQL server and any other OLE-compliant databases. Figure 3 illustrates the ADO .NET architecture.

It will also be possible to support non OLE-compliant databases by defining our own native implementation of these classes. However, due to the scope limitation of the current project, we will not be able to support this type of legacy databases.

Figure 2. System Architecture

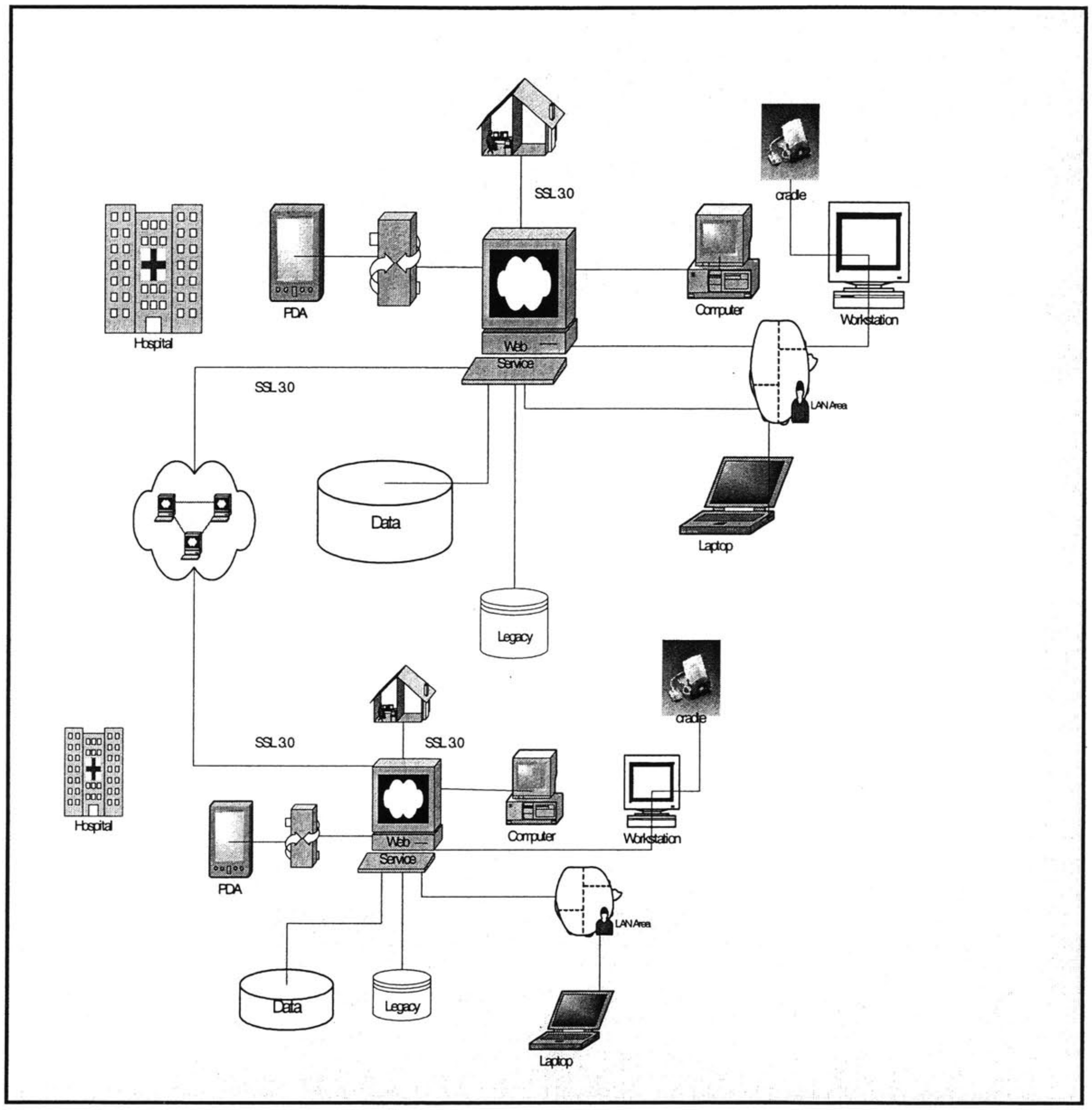
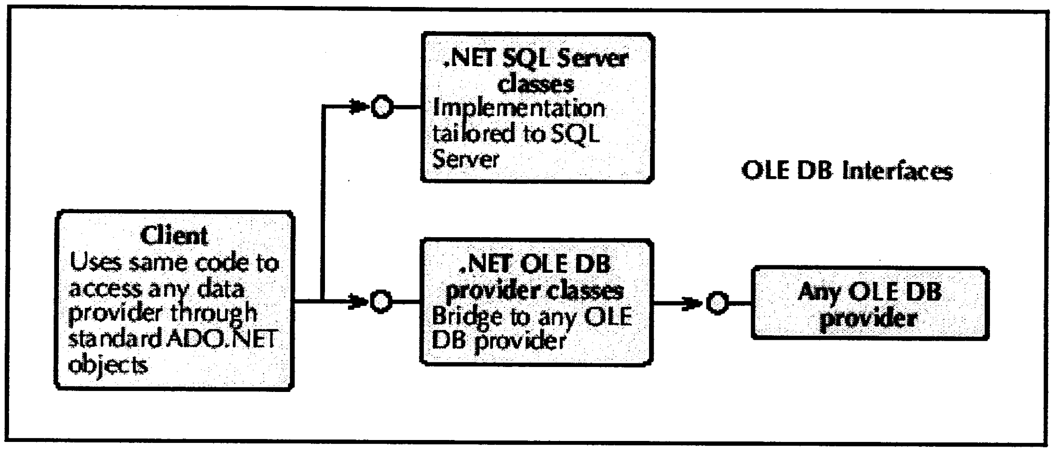


Figure 3. ADO.NET Architecture (taken from Platt 2001)


.NET SQL Server classes
Implementation tailored to SQL Server
OLE DB Interfaces
Client
Uses same code to access any data provider through standard ADO.NET objects
.NET OLE DB provider classes
Bridge to any OLE DB provider
Any OLE DB provider

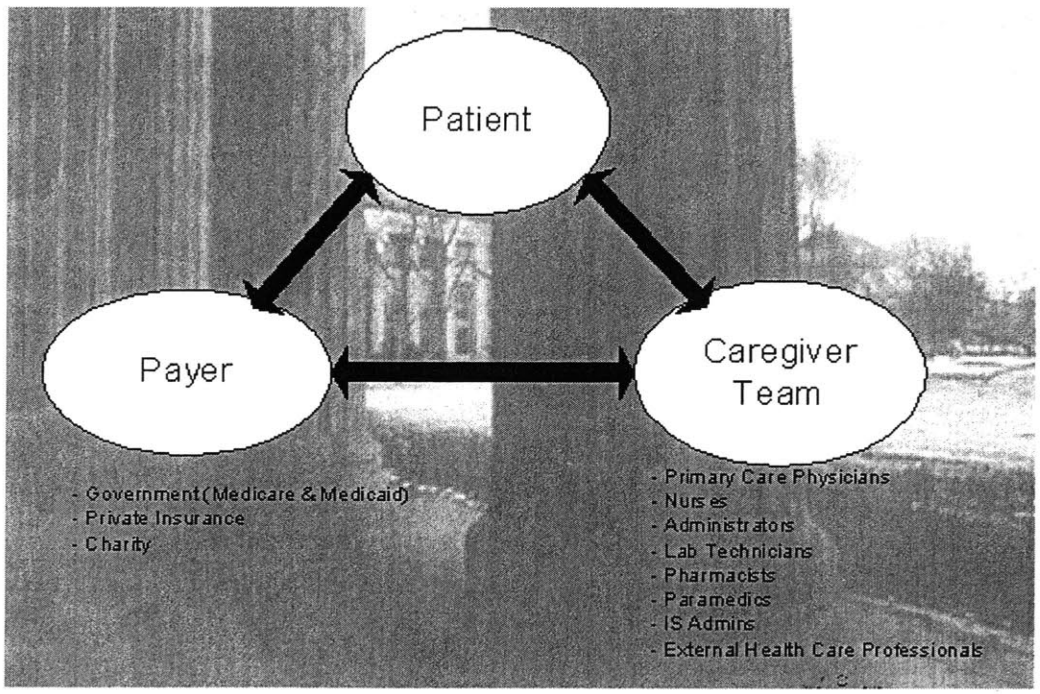
The web service is accessed by local clients (through intranet) as well as remote clients (through internet). These client applications will be built by creating .NET web applications. Notice that with the .NET technology, we will be able to allow client applications to consume web services from different sources. For example, the client for physicians will give them access to the internal patient database as well as external anonymous statistical information. On the other hand, the client for the patients will only give them access to the internal database.

Currently our architecture allows for both wired and wireless connections. Wireless connection will be very useful for mobile devices such as PDA's. The goal is to allow doctors and nurses to have patients' information with them when them make rounds and input data when they are away from their computers. Another important goal of using wireless technology is for paramedics to start taking down patient's personal information and some preliminary assessment in ambulances so that patient with urgent conditions can receive immediate treatment as soon as they arrive at the hospitals, since their data will already be entered.

However, given our time constraints and the complicated issues of how to ensure access to the web service when ambulances go out of the network, we might only implement a limited set of features in the wireless implementation. For the features not yet available, the doctors, nurses and paramedics can enter data off-line and later have their PDA's synchronized with a workstation through a cradle.

Figure 4 illustrates the three major user groups of our product and their interaction with each other.

Figure 4. Relationship Map



Patients are looking for better communications with their primary care physicians, easy access to information, better channels for self-education and a sense of involvement in decision-making regarding their health.

Physicians, which is the most representative user group in the caregiver team, look for ways to be time-efficient while improving quality of the health care they provide. They also look for tools that help them reduce medical errors such as typos or misreading of handwriting, etc. In addition, they need to be HIPPA compliant.

Hospitals and payers (private insurance companies and government health care programs such as MediCare), on the other hand, look for ways to save money and to use the resources smartly.

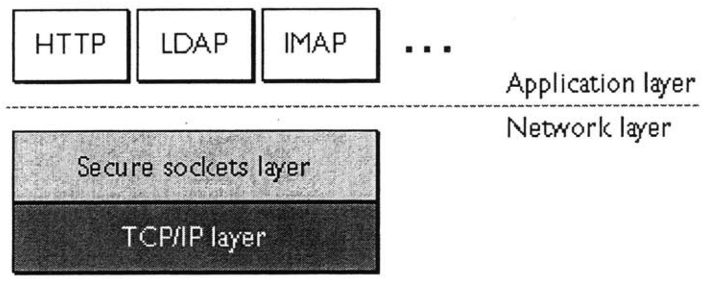
The solution we are designing is thus geared toward solving problems for these three groups of major players. Our system will allow easy access to data, thus saving time in chart-pulling and facilitate better workflow. We avoid medical errors by providing tools with drop-down menus for health care professionals to select drugs they prescribe, the diseases they diagnose and so on. Patients also have access to their own medical data, have tools to help them monitor their health so they can be taken care in a timely manner.

Finally, to ensure security in our system, since ASP.NET does not provide data encryption, we encrypt data by using the Secure Socket Layer 3.0 in the browser/server data transmission.

SSL was originally developed by Netscape and has been universally accepted on the World Wide Web for authenticated and encrypted communication between clients and servers. A draft of the SSL 3.0 Specification can be found at http://www.netscape.com/eng/ssl3/draft302.txt. The following is taken from the site http://developer.netscape.com/docs/manuals/security/sslin/contents.htm to provide an overview of the SSL protocol. Please refer to the website for more information.

"The Transmission Control Protocol/Internet Protocol (TCP/IP) governs the transport and routing of data over the Internet. Other protocols, such as the HyperText Transport Protocol (HTTP), Lightweight Directory Access Protocol (LDAP), or Internet Messaging Access Protocol (IMAP), run "on top of" TCP/IP in the sense that they all use TCP/IP to support typical application tasks such as displaying web pages or running email servers.

Figure 5 SSL runs above TCP/IP and below high-level application protocols (source: http://developer.netscape.com/docs/manuals/security/sslin/contents.htm)



The SSL protocol runs above TCP/IP and below higher-level protocols such as HTTP or IMAP. It uses TCP/IP on behalf of the higher-level protocols, and in the process allows an SSL-enabled server to authenticate itself to an SSL-enabled client, allows the client to authenticate itself to the server, and allows both machines to establish an encrypted connection.

These capabilities address fundamental concerns about communication over the Internet and other TCP/IP networks:

- **SSL server authentication** allows a user to confirm a server's identity. SSL-enabled client software can use standard techniques of public-key cryptography to check that a server's certificate and public ID are valid and have been issued by a certificate authority (CA) listed in the client's list of trusted CAs. This confirmation might be important if the user, for example, is sending a credit card number over the network and wants to check the receiving server's identity.

- **SSL client authentication** allows a server to confirm a user's identity. Using the same techniques as those used for server authentication, SSL-enabled server software can check that a client's certificate and public ID are valid and have been issued by a certificate authority (CA) listed in the server's list of trusted CAs. This confirmation might be important if the server, for example, is a bank sending confidential financial information to a customer and wants to check the recipient's identity.

- **An encrypted SSL connection** requires all information sent between a client and a server to be encrypted by the sending software and decrypted by the receiving software, thus providing a high degree of confidentiality. Confidentiality is important for both parties to any private transaction. In addition, all data sent over an encrypted SSL connection is protected with a mechanism for detecting tampering--that is, for automatically determining whether the data has been altered in transit.

The SSL protocol includes two sub-protocols: the SSL record protocol and the SSL handshake protocol. The SSL record protocol defines the format used to transmit data. The SSL handshake protocol involves using the SSL record protocol to exchange a series of messages between an SSL-enabled server and an SSL-enabled client when they first establish an SSL connection. This exchange of messages is designed to facilitate the following actions:

- Authenticate the server to the client.
- Allow the client and server to select the cryptographic algorithms, or ciphers, that they both support.
- Optionally authenticate the client to the server.
- Use public-key encryption techniques to generate shared secrets.
- Establish an encrypted SSL connection."

3.2 Design of Application Logic

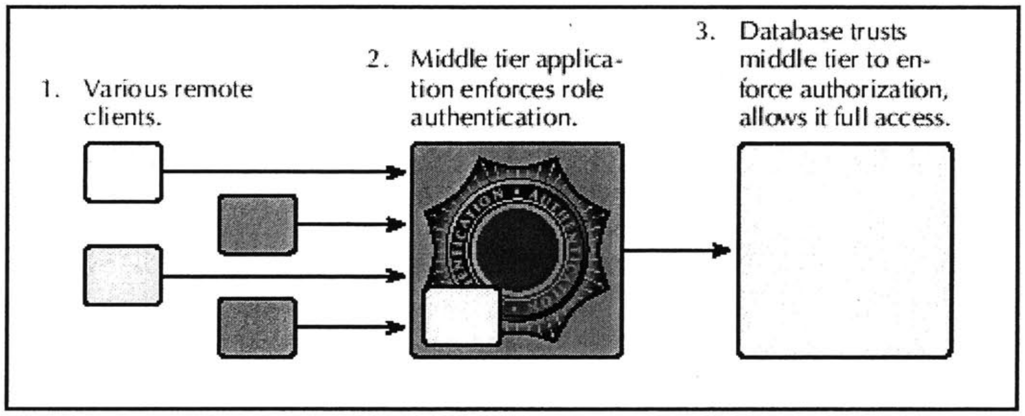
Appendix B outlines the design of our application logic. When a user logs in, their credentials are checked. If the user belongs to a certain role/group, e.g. Patients, Nurses, etc., an instance of that class is created for that particular user. This is known as the Role-based model. Essentially users have no individual identity outside of their role/group identity.

Each instance comes with certain methods available only to that specific role/group. This is especially crucial for a trusted user security model (see below for more details) we are adopting. The key here is to ensure that the user will never issue requests they do not have rights to. For example, in the Nurse class, only createProfile() and updateProfile() methods are available. Internal Physicians, on the other hand, have additional methods such as createMedicalHistory() and updateMedicalHistory(). We do not want nurses to be able to create or update patients' medical history. With this application logic, a user who belongs to the group Nurse will never be able to call createMedicalHistory() or updateMedicalHistory() since these methods are undefined for their class.

When a user does issue requests they have rights to, the Controller interface checks for further details such as what rights this Role/Group has to certain tables. In the case of physician ordering tests or pharmacists filling prescriptions, the Controller checks to see whether an electronic signature is attached and whether it is a valid signature file for the requestor and the requested record (this last part involves more than just role/group identity).

Once the controller verifies the validity of the requests, the corresponding stored procedures are executed. No additional security check is done on the database side, as we have made sure only valid requests would come through. This is known as the Trusted User Model, as illustrated in Figure 6 below.

Figure 6. Trusted User Model (taken from Platt 2001)


1. Various remote clients.
2. Middle tier application enforces role authentication.
3. Database trusts middle tier to enforce authorization, allows it full access.

3.3 Database Modeling

We have considered various possibilities for our database system, including the traditional relational database management system (RDBMS), object relational database management system (ORDBMS), object database management system (ODBMS) and XML datastore.

As we would like to incorporate legacy database systems, which are mostly relational, we are leaning toward either an RDBMS or an ORDBMS system. There are other advantages as well. For one thing, querying an RDBMS or an ORDBMS system is significantly faster. However, since patient files contain objects such as pictures, charts, X-rays, audio files (for example, session tapes for psychiatric patients), electronic signature files, etc, it makes more sense to adopt an ORDBMS system.

ORDBMS is essentially a relational database with object storage. Objects are either stored directly in columns within a relational database table, or references to objects are stored. The fact that the database is presented in terms of tables and views makes it compatible with relational databases. Thus, we can store objects while working with legacy relational database systems.

We have chosen to use Oracle 8.1.7 as our database server. Oracle 8.1.7 provides and supports both relational and object-relational data models. Appendix C is the initial data model of our database. We will be able to store objects and nested tables in a relational database.

A brief note on the XML datastore. We have chosen to go with a more traditional database system mainly to avoid unnecessary learning curves in adopting new technology, where the use of such technology has not yet been standardized and the relevant documentation is not easily accessible. Nonetheless, we have requested a copy of the Tamino XML Server (from Software AG, the XML Company), which shows great potentials.

3.4 User Case and Scenarios

Finally, Appendix D lists some of the use cases and scenarios we considered in the original design.

4 Final Implementation

Section 3 System Architecture and Design outlined the original design we envisioned. This section discusses the design changes we have made and the factors and reasoning that weighed in and led to these decisions. Compromises and constant adjustments are an inevitable part of software development, as unexpected surprises come up and circumstances change. It is essential to learn how to modify current schedule accordingly and make the necessary adjustments in a timely manner.

4.1 Time Constraints

Time constraints have the most significant impacts on our design changes, as we have committed ourselves to the project deadline right from the beginning, i.e. date-driven commitment.

The initial design was completed in January, which left us three months to code, test and integrate. Even though .NET provides us with the Visual Basic 'Drag and Drop' feature, which allows easy creation of controls and which significantly simplifies the coding process, we still needed to be careful in making the best use of the limited amount of time we have.

Thus, we avoided unnecessary learning curves in adopting new technologies such as Tamino XML server and Oracle 9i (more discussions below). In addition, in order to make sure we will deliver on time, we have chosen to implement a user interface and client for patients and another interface plus client for the physicians only. The need to implement the UI and client for patients in the first deliverable requires no justification, as they are one of the most important user groups of the product. We had planned to have additional UIs and clients for each role who is granted access to the database: nurses, paramedics, lab testers, HMO, administrators, IS administrators and external health care professionals. However, we consider physicians the most representative role among all the health care professionals, as they have the most interaction with patients. By implementing UI and the client first for the physicians and patients, we would be able to get first-hand and valuable input from them, if we were to deploy our product in stages. The UIs and clients for other health care professionals are relatively easy to extend from the existing product.

4.2 Resource Constraints

An additional limitation we experienced is insufficient access to crucial resources, which includes finding out how database management systems are currently implemented, utilized and maintained in the health care industry (our goal is to have our product integrated into the existing health care system seamlessly and thus it is essential to have a clear picture of what the existing systems look like that we are integrating into). Ideally we would have liked to be able to consult with IS administrators in several hospitals and clinics to get an idea about these systems. However, in addition to time constraints, we also did not have access to these professionals and we did not have enough human resources to conduct these researches, either.

In addition, we did not have direct contact to HMO health insurance carriers, which are also important players in the health care industry. Since they will be utilizing the health-related data in different ways from health care professionals, say, physicians, we also need to seek their input and make sure the product we are developing fits their needs and helps them achieve their goals.

Furthermore, since none of our team members has background in medicine, our knowledge about medical products and terminology is very limited. We thus have chosen to implement two diseases only: heart disease and diabetes. It would have been extremely helpful if we could have been able to recruit a medical consultant who had been involved in the database modeling process to make sure we include all the necessary data fields in the tables, their formats and validation checks.

Resource constraints such as these have forced us to make several assumptions about what our customers' needs are, which will crucially need a thorough reality check.

4.3 Legal Issues

Even though our primary target customers are health care professionals and patients, we do not have access to actual medical data from hospitals and clinics, as there are legal issues involved, which are out of the scope of the project. Thus, we had to come up with limited amount of test data that might not be compatible with real data. In addition, data mining, i.e. using anonymous statistical data to aid physicians in decision-making, is hard to implement and relatively unrealistic without sufficient actual medical records.

Furthermore, we have adopted the localized model in terms of database ownership. The reasoning is mainly to simplify our product architecture. However, we do not have sufficient legal knowledge to know whether such assumptions are realistic. Again, we did not have legal resources to ensure that our product indeed conforms to all legal requirements and realities.

4.4 Technology Compatibility

We experienced some difficulties regarding compatibility issues among various technologies we have employed. For example, we had installed Oracle Enterprise Edition 8.1.7 on Pentium 4 machines, only to realize that they are not compatible, i.e. Oracle 8i is not supported on Pentium 4 machines. There are further issues with our operating system being XP not compatible with Oracle 8i. We managed to secure software for Oracle 9i afterwards. However, given that Oracle 9i has just been released recently and thus the documentation and resources are not sufficient or useful, especially since we are also adopting .NET, which is also emerging technology, the risks of further incompatibility seemed to us to be too high. We thus opted to go with Microsoft SQL server, which was designed to work well in the .NET environment, even though the functionality seems to be limited compared to Oracle or DB2.

Another reason that we switched to Microsoft SQL server is due to easier manageability. It is extremely difficult to maintain and administer a database such as Oracle or DB2 without sufficient training in such areas, as these systems are very complicated. Even though we have managed to figure out the fundamentals for Oracle 8i, we felt it would be better to adopt a database system that is easier to maintain.

4.5 Legacy Systems

Currently our product is shipped with a built-in database for each hospital, which will be maintained by hospital IS administrators. This database can be pre-configured to suit individual hospitals' needs. But in any case, the database is built from scratch. However, we definitely intend to take advantage of the existing database systems the hospitals are already employing and maintaining and it is to be expected that most hospitals already maintain some kind of database for at least administrative purposes, e.g. patient profiles, and to some degree for keeping data containing medical diagnoses and medical histories.

.NET is designed to support all OLE-compliant databases and it is possible to support non OLE-compliant databases by defining our own native implementation. However, due to time constraints, we have not attempted to generate code that will eventually support or transfer data from legacy database systems. This aspect will definitely be a deciding factor if our product is to be marketable.

4.6 Debugging and Testing Issues

One of the initial reasons that we decided to go with MS SQL server instead of Oracle9i is because of the complexity of administrating the new Oracle 9i system and lack of useful documentation. More importantly, the fact that Oracle does not seem to provide a user-friendly front end for accessing the back end data complicated things further. We basically created all the roles, tables, etc. in the DOS command windows, which is inefficient and time-consuming.

Even though we were aware of the fact that it is possible to have MS Access as a front end to Oracle 8i, we were not sure whether it would work just as smoothly with Oracle 9i. In addition, we felt it would unnecessarily complicate debugging, i.e. if we are having trouble with the database, we need to figure out first of all whether it is a problem with MS Access, with Oracle, or interaction between the two. We thus opted to go with MS SQL server, which is both a front end for displaying data and a back end server for storing data. We felt it is necessary to simplify the debugging process given the limited amount of time we have.

4.7 Database Issues

Another simplification we made in implementing the original design is to adopt a predominantly relational database model rather than the object-relational database model we designed. The reason for this is related to the resource and legal constraints in that we do not have access to actual medical data, in particular binary data such as X-rays, session transcriptions, etc. Thus, in the actual implementation, we have been using test data that we generated, which are mostly texts. This also simplifies testing in the initial implementation stage.

5 Evaluation of the Project and Concluding Remarks

In sum, even though our solution has exciting potentials in improving the quality of health care and work efficiency in the health care industry in an innovative way, we had to simplify several aspects of the original design in our final implementation to make sure we meet our date-driven commitment. These simplifications include:

- reduction of the feature requirements, e.g. only patients and physicians currently have an UI to access the web service,

- intuitive and simplistic assumptions about legal issues, i.e. adopting the Localized model for database ownership,

- simplification of the medical resources, i.e. implementing two diseases only: heart disease and diabetes,

- simplification of the database structure, i.e. relational database model which does not contain binary data formats such as images, videos, etc.,

- legacy database systems, non-OLE compliant database systems, i.e. not currently supported

Figure 7 below shows in black solid lines the parts of the design we were able to implement. The red dotted lines show that parts that we were not able to implement due to the constraints discussed earlier. Basically we have left out interactions with legacy database systems and so far different hospitals cannot communicate with each other yet. SSL has not been implemented, either, which is shown in strikethroughs.

Figure 7 System Architecture – Final Implementation

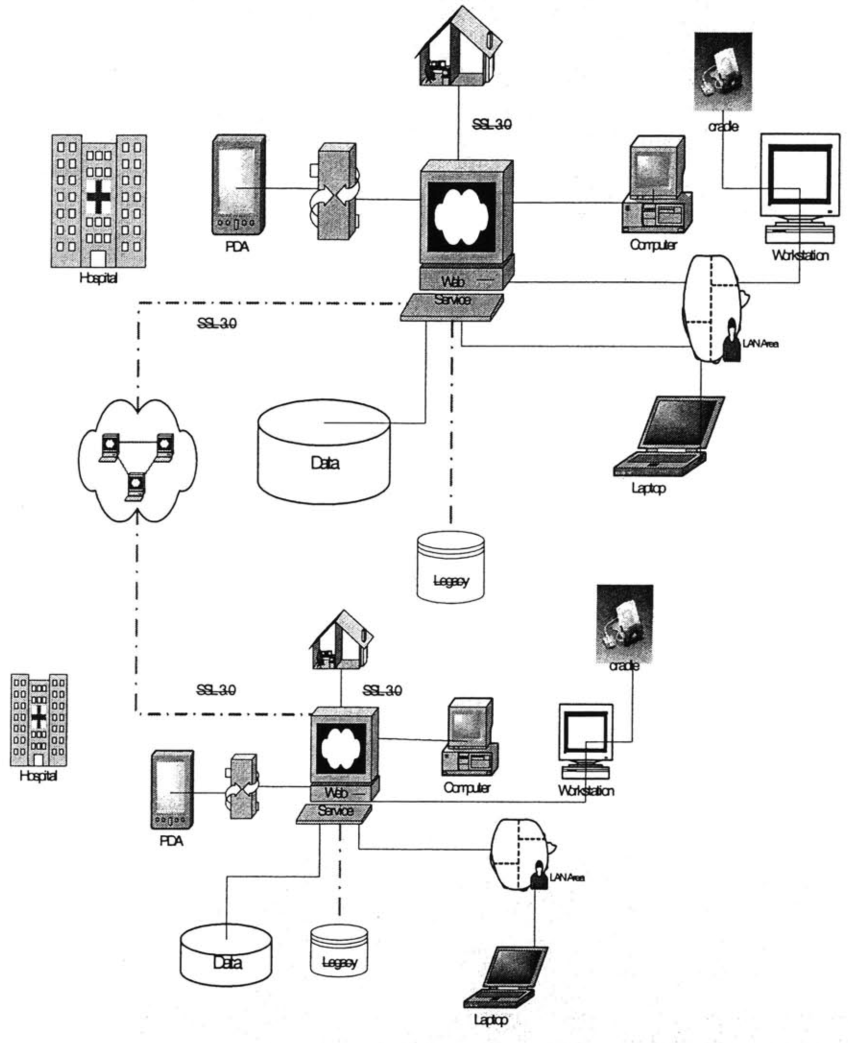
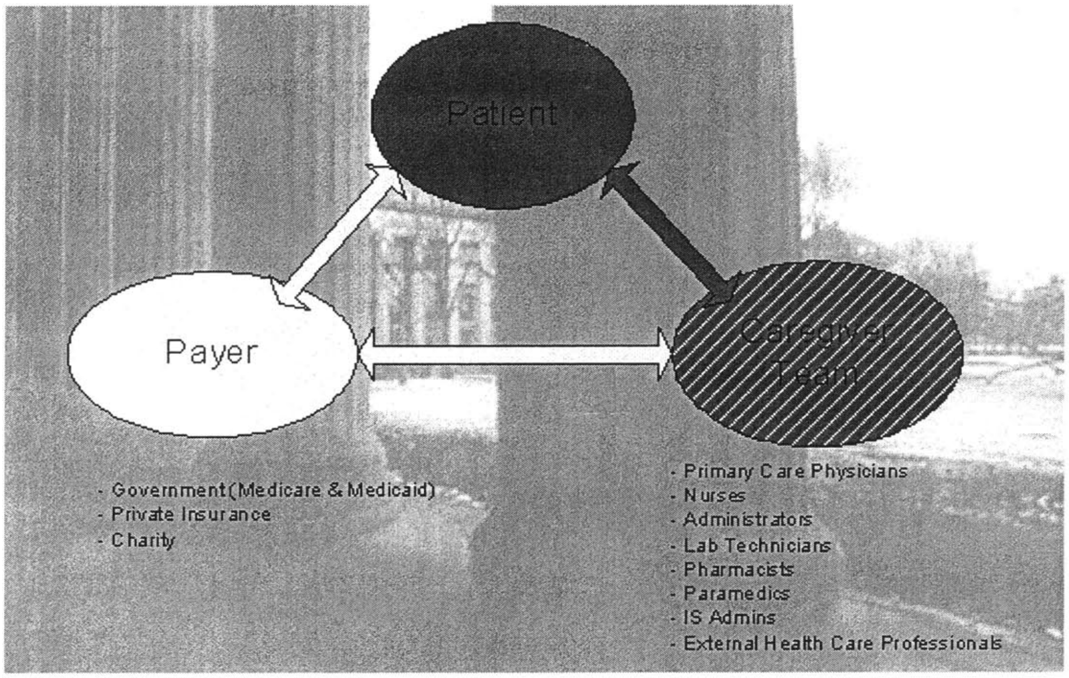


Figure 8 shows the parts of the relationship map we were able to implement, i.e. patients and their primary care physicians, who have the most interaction with the patients, and the interaction between these two groups.

Figure 8 Relationship Map – Final Implementation



It is thus perhaps more appropriate to think of our first deliverable as a prototype which contains the basic and indispensable features. We could then present this prototype to potential customers and assure them the extensibility of our product and our determination to implement the complete/target feature specifications according to their needs. In the meantime, we will start gathering their input on the existing prototype so we can keep improving the quality of our product.

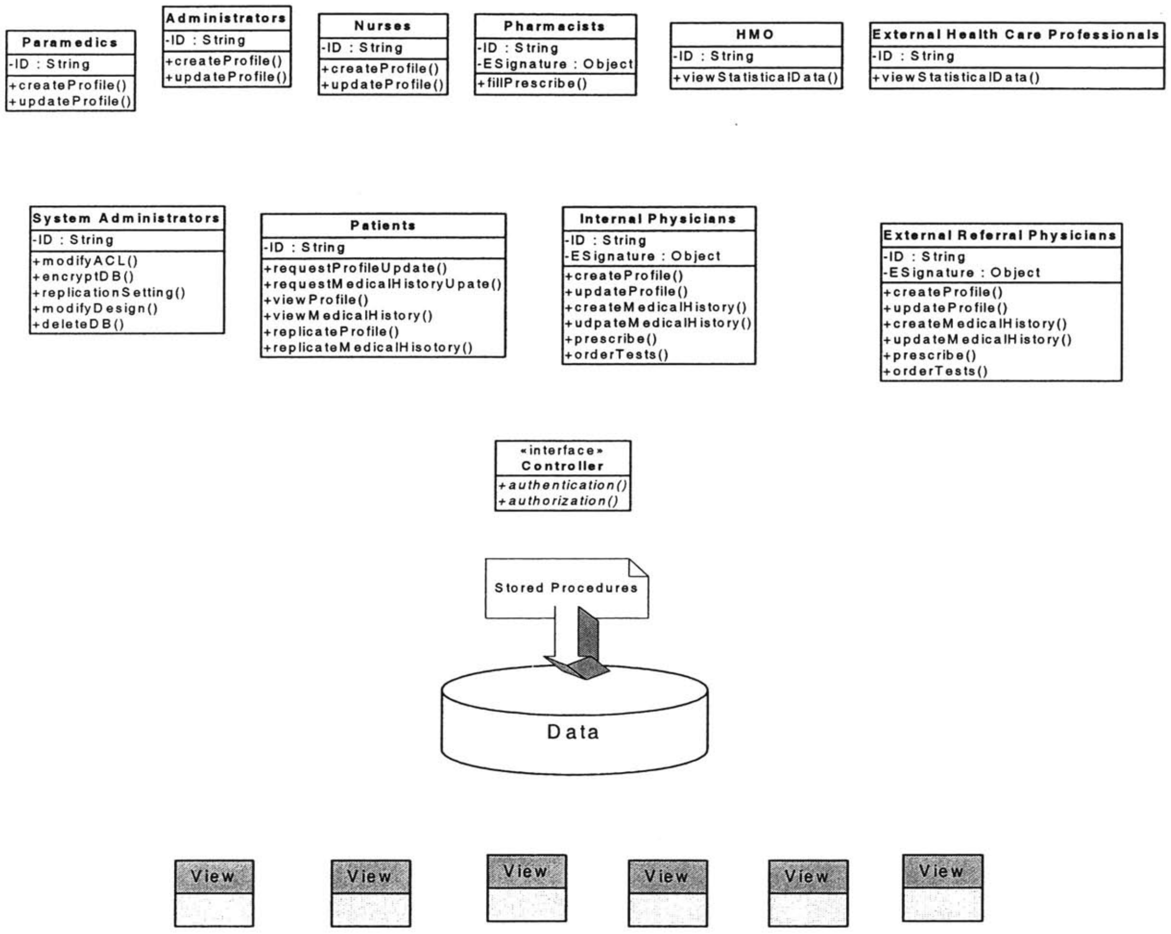
In order to make our product more marketable, we need to

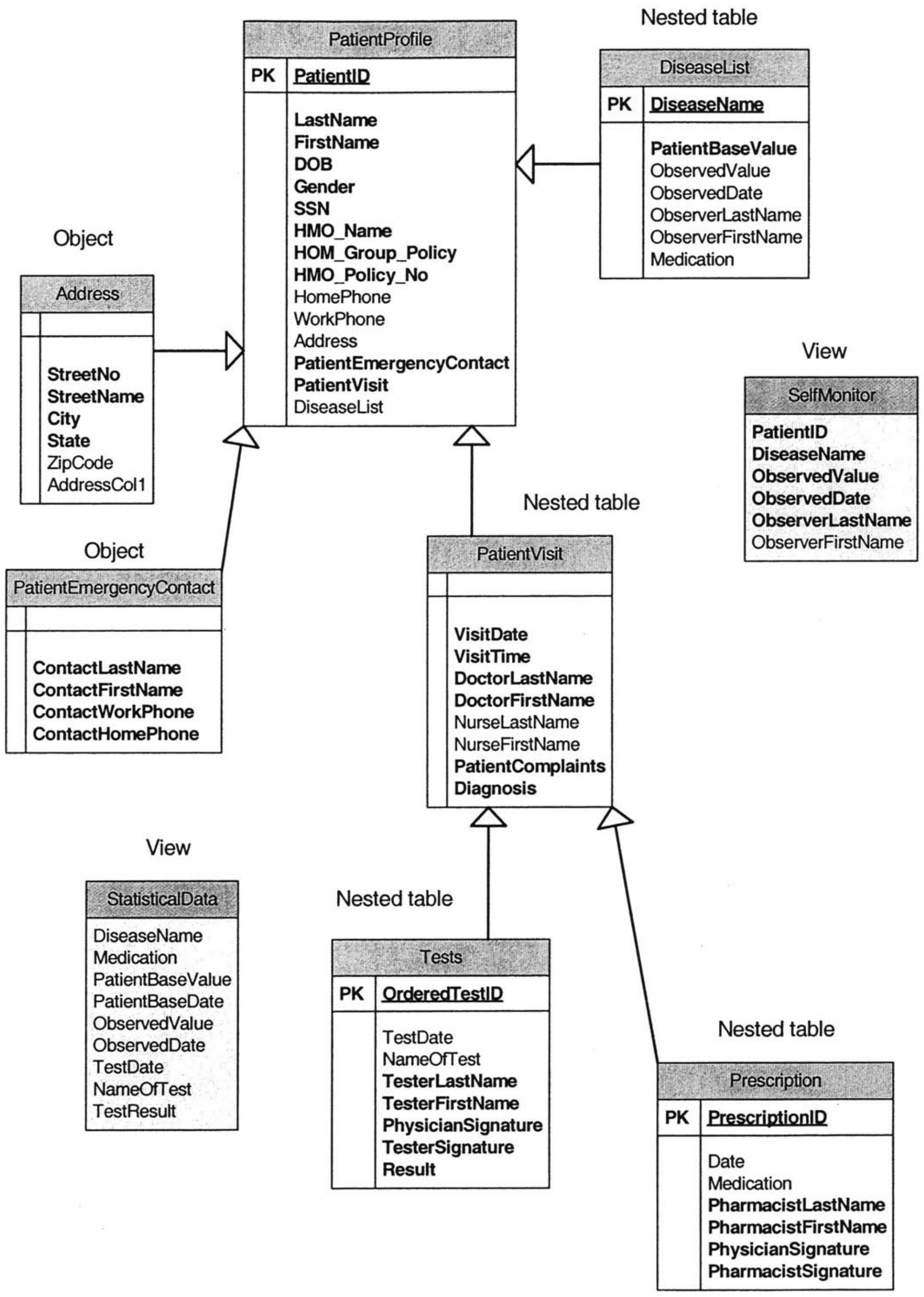
- extend our product to cover all user groups
- have wider coverage of medical resources that will be needed in our web service
- provide support for object storage in the database
- provide support for legacy database systems
- do a reality check on database ownership and legal issues and subsequently modify the architecture, design and implementation accordingly
- include relevant experts and players (hospital IS consultants, HMO, medical experts, legal consultants, patients, physicians, etc.) in the evaluation process
- in addition, we need to implement SSL, which due to lack of time, was left out, to ensure data security
- finally, for the sake of better design and future maintainability of code, we need to transfer SQL commands which are currently hard-coded in individual web methods into Stored Procedures.

Appendix A. Access Control List

Table / Roles	Patient Profile	Patient Emergency Contact	Patient Visit	Disease List	Self Monitor	Prescription	Tests	Statistical Data
Patients	SELECT UPDATE (HomePhone, WorkPhone, StreetNo, StreetName, City, State, Zipcode)	SELECT UPDATE (ContactLastName, ContactFirstName, ContactWorkPhone, ContactHomePhone)	SELECT	SELECT	SELECT INSERT	SELECT	SELECT	No access
Internal Physicians	SELECT UPDATE INSERT	SELECT UPDATE INSERT	SELECT UPDATE INSERT	SELECT UPDATE INSERT	SELECT	SELECT UPDATE INSERT	SELECT UPDATE INSERT	SELECT
Internal Nurses	SELECT UPDATE INSERT	SELECT UPDATE INSERT	SELECT UPDATE INSERT	SELECT UPDATE INSERT	SELECT	SELECT	SELECT	SELECT
Paramedics	SELECT UPDATE INSERT	SELECT UPDATE INSERT	No access	No access	No access	No access	No access	No access
Pharmacists	SELECT	SELECT	SELECT	SELECT	SELECT	SELECT UPDATE	SELECT	SELECT
Testers	SELECT	SELECT	SELECT	SELECT	SELECT	SELECT	SELECT UPDATE	SELECT
HMO	SELECT UPDATE	SELECT	SELECT	SELECT	No access	SELECT	SELECT	SELECT
Administrators	SELECT UPDATE INSERT	SELECT UPDATE INSERT	SELECT UPDATE (VisitDate, VisitTime, DoctorLastName, DoctorFirstName, NurseLastName, NurseFirstName)	No access	No access	No access	No access	No access
IS Admins	SELECT UPDATE INSERT DELETE	SELECT UPDATE INSERT DELETE	SELECT UPDATE INSERT DELETE	SELECT UPDATE INSERT DELETE	SELECT UPDATE INSERT DELETE	SELECT UPDATE INSERT DELETE	SELECT UPDATE INSERT DELETE	SELECT UPDATE INSERT DELETE
External Referral Physicians	SELECT	SELECT	SELECT INSERT	SELECT INSERT	SELECT	SELECT INSERT	SELECT INSERT	SELECT
External Health Care Professionals	No access	No access	No access	No access	No access	No access	No access	SELECT

Appendix B. Application Logic and Class Diagrams




Nested table
PatientProfile
PK PatientID
LastName
FirstName
DOB
Gender
SSN
HMO_Name
HOM_Group_Policy
HMO_Policy_No
HomePhone
WorkPhone
Address
PatientEmergencyContact
PatientVisit
DiseaseList
DiseaseList
PK DiseaseName
PatientBaseValue
ObservedValue
ObservedDate
ObserverLastName
ObserverFirstName
Medication
Object
Address
StreetNo
StreetName
City
State
ZipCode
AddressCol1
View
SelfMonitor
PatientID
DiseaseName
ObservedValue
ObservedDate
ObserverLastName
ObserverFirstName
Nested table
Object
PatientEmergencyContact
ContactLastName
ContactFirstName
ContactWorkPhone
ContactHomePhone
PatientVisit
VisitDate
VisitTime
DoctorLastName
DoctorFirstName
NurseLastName
NurseFirstName
PatientComplaints
Diagnosis
View
StatisticalData
DiseaseName
Medication
PatientBaseValue
PatientBaseDate
ObservedValue
ObservedDate
TestDate
NameOfTest
TestResult
Nested table
Tests
PK OrderedTestID
TestDate
NameOfTest
TesterLastName
TesterFirstName
PhysicianSignature
TesterSignature
Result
Nested table
Prescription
PK PrescriptionID
Date
Medication
PharmacistLastName
PharmacistFirstName
PhysicianSignature
PharmacistSignature

Name: Lab Test Reports

Description: Doctors/Nurses may print a list of lab tests available in each laboratory for a particular patient. In addition basic information of each test may be viewed or printed. They may also review/print statistical information of tests performed.

Name: Make Lab Order

Description: Doctors/Nurses can make lab orders for the patient. The priority of the order may also be defined.

Postconditions: The lab order is visible in laboratory after the order has been added to the system. The lab order may be queued. Doctors/Nurses and lab users can follow the progression of the lab process.

Exception1: Lab worker may have marked some lab tests temporarily unavailable on his unit. The system then suggest another unit to the Doctors/Nurses user, if possible.

Exception2: Doctors/Nurses user may cancel the lab order before its completion.

Further development: The system or laboratory manager may make schedules for ordered tests, and the patient may be provided with a note of the time and place of the test. Also a interface to resource planning system of the laboratory may be provided.

Component: Lab process

Name: Perform lab test

Description: Lab worker encounters the patient and performs the test which has been assigned to him. The sample may then be assigned with an analyzer or the lab worker may analyze the sample himself.

Preconditions: Lab test has been assigned with a performer and time according to "Receive lab order" use case.

Postconditions: The lab test has been performed and the sample in sent for analysis.

Exception1: It is possible to cancel lab order at this point, for example if the patient does not show up.

Further development: The system may print stickers containing the patient and test information for analysis.

Component: Lab process

Name: Patient reports

Description: Doctors/Nurses can view and print a *list of patients on a ward.* Doctors/Nurses or lab worker can print *personal and lab information of a patient* to be taken care of. Doctors/Nurses user can also print a *list of lab results of patients on the ward for selected date(s).*

Name: Receive lab order

Description: Lab user takes lab orders from the queue, according to its priority, confirms the availability of needed resources and assigns time and performer of the test.

Preconditions: The lab order has been added to the system according to use case ”Make lab order”.

Postconditions: The lab order has been assigned with the lab worker to perform the test and time.

Exception1: If all resources are not available, it is possible to transfer lab order to different laboratory or unit which is able to complete the test.

Further development: See ”Make lab order”, further development

Component: Lab process

Name: Register lab result

Description: Lab user registers the lab result of a patient to the system. After the result has been registered, the department and Doctors/Nurses user responsible of the lab order are notified and can view the lab result.

Preconditions: The patient has been tested according to ”Perform lab test” use case, the specimen has been analyzed and results are available.

Postconditions: The lab result of the patient has been registered to the system and lab process has been completed.

Component: Lab process

Name: Update lab tests

Description: Lab user may insert and modify lab tests and their descriptions. He can also assign lab tests for his unit and mark tests of his unit temporarily unavailable.

Component: Lab test

Name: Update Profile

Description: Doctors/Nurses user can insert new people to the system, update patients' personal information and assign patients to a ward. Patients are assigned to at most one ward at a time. Patient can also update their personal information.

Preconditions: Patient information can be viewed by an employee who is participating in the care on a unit or a laboratory. Patient information can be changed by participating Doctors/Nurses user or by patient himselp

Further development: Patient transfers, admissions, discharges.

Component: Person

Name: View lab results

Description: Doctors/Nurses user providing care for the patient may view and print lab results of the patient.

Preconditions: The test result has been added to the system according to "Register lab result" use case.

Component: Lab process

REFERENCES

Matheson, David, and Tom Robinson. 2000. "Disease Management Takes Flight." The Boston Consulting Group Health Care Practice. 22 Oct.2001.<http://www.bcg.com/publications/search_view_ofas.asp?pubID=570>.

Platt, David. 2001. Introducing Microsoft .NET. MS Press.