IMPROVING ACCESS TO DATA IN LEGACY HEALTH INFORMATION SYSTEMS

by

Fernando Belli
B.Sc. Major Computer Science, University of Calgary, 2004
B.Sc. Minor Applied Mathematics, University of Calgary, 2004

PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

In the
Department of Computing Science

© Fernando Belli 2009
SIMON FRASER UNIVERSITY
Fall 2009

All rights reserved. However, in accordance with the Copyright Act of Canada, this work may be reproduced, without authorization, under the conditions for Fair Dealing. Therefore, limited reproduction of this work for the purposes of private study, research, criticism, review and news reporting is likely to be in accordance with the law, particularly if cited appropriately.
APPROVAL

Name: Fernando Belli
Degree: M.Sc. Computing Science
Title of Thesis: IMPROVING ACCESS TO DATA IN LEGACY HEALTH INFORMATION SYSTEMS

Examining Committee:
Chair: Dr. Steven Pearce
Lecturer

____________________________________
Dr. Stella Atkins
Senior Supervisor
Professor Emeritus

____________________________________
Gregory Baker
Supervisor
Senior Lecturer

____________________________________
Dr. Wo-Shun Luk
Internal Examiner
Professor

Date Defended/Approved: December 7th, 2009
Declaration of Partial Copyright Licence

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection (currently available to the public at the “Institutional Repository” link of the SFU Library website <www.lib.sfu.ca> at: <http://ir.lib.sfu.ca/handle/1892/112>) and, without changing the content, to translate the thesis/project or extended essays, if technically possible, to any medium or format for the purpose of preservation of the digital work.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author’s written permission.

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

While licensing SFU to permit the above uses, the author retains copyright in the thesis, project or extended essays, including the right to change the work for subsequent purposes, including editing and publishing the work in whole or in part, and licensing other parties, as the author may desire.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

Simon Fraser University Library
Burnaby, BC, Canada
ABSTRACT

In today’s world, fast and on time access to information has become a necessity due to the increasing demand to make informed decisions based not only on estimations and historical information but also on current data. Decision makers expect to have access to the information in a timely way. This is a particular problem in the Fraser Health Authority where data needed by decision-makers is not easily accessible due to lack of functionality in their legacy Health Information Systems (HIS). Our solution involves extracting the data from the HIS into a SQL database every 15 minutes, that is then used to create a set of XML files. These XML files are processed and displayed to the end users using a client-side browser application in a secure way. This solution adheres to the policies of the Fraser Health Authority, is scalable for read-only access, and enables end-users to access current information.

Keywords: Health Information Systems; Health Care Operational Dashboards; Meditech Systems
ACKNOWLEDGEMENTS

A special thank you to Dr. Stella Atkins, my senior supervisor, for her continued support, encouragement, and guidance during my studies and the development of this project. It has been a great experience working under her supervision and to be able to gain from her knowledge and experience.

Also my gratitude to Gregory Baker, my supervisor, for his participation and advice in the project. To Dr. Wo-Shun Luk for agreeing to examine the project and for his comments. To Glenn Weigel and Abelardo Mayoral for their contributions in the design of the prototype and for making possible its implementation. And to the administrative staff at the Abbotsford Regional Hospital and Cancer Centre for their willingness to participate and contribute to the development of the prototype.
# TABLE OF CONTENTS

Approval.............................................................................................................................................. ii
Abstract.................................................................................................................................................. iii
Acknowledgements.............................................................................................................................. iv
Table of Contents................................................................................................................................ v
List of Figures........................................................................................................................................ vii
List of Tables.......................................................................................................................................... viii
Glossary.................................................................................................................................................... ix

## 1: Introduction
1.1 Objective.......................................................................................................................................... 1
1.2 Background....................................................................................................................................... 2
   1.2.1 Fraser Health Authority (FHA)............................................................................................... 2
   1.2.2 Medical Information Technology, Inc. (Meditech)................................................................. 3
   1.2.3 Health Information Systems (HIS).......................................................................................... 4

## 2: Requirements
2.1 User’s Requirements....................................................................................................................... 6
2.2 Organization’s Requirements.......................................................................................................... 8

## 3: Data Sources
3.1 Meditech Systems........................................................................................................................... 9
3.2 Other Sources................................................................................................................................ 12

## 4: Research
4.1 Data Extraction............................................................................................................................... 14
   4.1.1 Meditech Custom-Made Interfaces.......................................................................................... 14
   4.1.2 Meditech Download Reports.................................................................................................. 14
   4.1.3 IPeople Connect.................................................................................................................... 16
4.2 Data Load and Storage.................................................................................................................. 18
4.3 Data Processing and Presentation................................................................................................. 20
   4.3.1 Architecture.......................................................................................................................... 20
   4.3.2 Programs............................................................................................................................... 22

## 5: Prototype
5.1 Design.............................................................................................................................................. 24
5.2 Implementation............................................................................................................................... 26
   5.2.1 Extraction Programs.............................................................................................................. 28
   5.2.2 Processing Program............................................................................................................... 30
   5.2.3 GUI and Presentation Program.............................................................................................. 30
5.3 Results............................................................................................................................................... 32
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Meditech Basic Architecture</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>IPeople Connect Architecture</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>Basic 3-Tier Architecture</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>3-Tier Data Processing and Presentation Architecture</td>
<td>22</td>
</tr>
<tr>
<td>5</td>
<td>Architecture of the Implemented Solution</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>Dashboard: Current Summary Screen</td>
<td>27</td>
</tr>
<tr>
<td>7</td>
<td>Dashboard: Current Emergencies Screen</td>
<td>27</td>
</tr>
<tr>
<td>8</td>
<td>Dashboard: Current Inpatients Screen</td>
<td>28</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 1 - User’s Requirements ............................................................................................................. 7
Table 2 - Meditech Data Sources .......................................................................................................... 12
GLOSSARY

AJAX  Asynchronous JavaScript and XML
FHA  Fraser Health Authority
GUI  Graphic User Interface
HIS  Health Information System
IHA  Interior Health Authority
ODBC  Open Database Connectivity
VBA  Visual Basic for Applications
VBS  Visual Basic Script
XML  Extensible Markup Language
1: INTRODUCTION

In today’s world, fast and on time access to information has become a necessity due to the increasing demand to make informed decisions based not only on estimations and historical information but also on current data. Decision makers expect to have access to the information they need when they need it. This is a trivial task when all the data they need is properly structured and stored in a single system that has the appropriate features to present it. The task becomes a lot more complex when the required information is stored in different systems and/or systems that lack the functionality to properly present it. In addition, existing policies and guidelines in the organization sometimes add to the complexity of the task since they might limit how the task could be done and what data access is permitted.

The project explores the possible ways that current data could be extracted and made available to decision makers from systems that lack the proper functionality to do so within the existing policies and guidelines of an organization. In particular, the project focuses in extracting and presenting current operational data from two Health Information Systems used by the Fraser Health Authority (FHA) in British Columbia.

Like any other modern organization, FHA’s management teams require access to current and accurate operational information to monitor the status and utilization of resources at each of its hospitals and the entire system as a whole. This in turn allows management to make informed decisions about how to best allocate their limited and scarce resources. One of the major challenges that the organization faces is that their current two Health Information Systems have very limited capabilities to provide most needed up-to-date information to their directors and managers in a friendly and easy to access format. Their systems are capable of providing information for each hospital or a group of hospitals through reports, but they do not have any capability at all to provide such information at the system level since there is not an application in place that combines the information of both systems. In addition, the reporting approach to present information is not totally user friendly and it lacks the ability for the users to be able to interact with the data in order to get the most of it in the fastest and more efficient way.
The project shows that it is possible for FHA to improve the access to the information held in their two HISs by developing a software solution that could be implemented in house, in a small period of time, using the existing set of software tools available in the organization, and following FHA’s existing guidelines and policies.

1.1 Objective

Our project explores the possible ways that current data could be extracted and made available to decision makers from systems that lack the proper functionality to do so in a real world setting. In particular, the project focuses on developing a prototype solution for extracting, processing, consolidating, and presenting current data from two Health Information Systems used by the Fraser Health Authority following the technical and policy constraints existing in the organization.

As mentioned before, FHA management teams require access to current and accurate operational information to run their day-to-day operations, so the proposed solution focuses on extracting and displaying this particular kind of data. The project currently focuses on the set of data that provides information about the current status of the hospitals and the entire system as a whole, with the possibility of expansion later on to look for trends over different periods of time.

1.2 Background

1.2.1 Fraser Health Authority (FHA)

The Fraser Health Authority was created in December 2001 from the amalgamation of three health authorities and became one of the six provincial health authorities that exists today in British Columbia. FHA provides integrated health care services to more than 1.5 million people along the Fraser Valley, between Burnaby, Hope, and the US border. FHA has over 23,000 employees and 2,300 physicians working across 12 hospitals and dozens of other care health facilities (Fraser Health Authority, 2009).
As a result of the amalgamation, FHA ended up with three HISs, which was not advantageous, however to their advantage, all of them were products developed by the same vendor, Medical Information Technology, Inc. (Meditech). As of March 2007, there are only two HISs in use in the organization: Meditech Magic, used primarily in four hospitals and a few long-term care facilities, and Meditech Client/Server used in the other eight hospitals and a handful of long-term care facilities. Both systems function independently of each other, there are not communication interfaces set between them to exchange data or to manage them as a single system. There are plans to upgrade and/or consolidate the Magic system to/with the Client/Server system, but due to the magnitude of the work and the amount of resources needed to do so, it is expected that the upgrade/consolidation will not be completed for at least a year or more.

Like any other health organization, FHA has very strict policies regarding the use and access to patient information, including administrative and medical information. All data containing detailed patient information has to be stored in secure databases or network drivers and access to it must be available only to its required audience. Development of applications using professional software tools is limited to a few departments inside the organization; the preferred procedure is to outsource the development of any applications to external companies or to buy entire solutions out of the box. The only exception to this guideline is the development of small applications using the features and functionality available in the applications installed in all computers across the organization. These kinds of applications are primarily intended to be used only by the departments that develop them and by their clients. Like any large public organization, changes to policies and/or guidelines as well as the approval for the development of new custom-made software solutions by external vendors could take a considerable amount of time and effort that could take a few months to a year or even more.

1.2.2 Medical Information Technology, Inc. (Meditech)

Meditech is a privately held US company with headquarters in Boston Massachusetts that has been developing, installing, and supporting HISs for 40 years. It has over 2,200 customers worldwide and employs more than 2,700 people (Medical Information Technology, Inc., 2009)
A characteristic of almost all Meditech products is that they were and are developed using proprietary technology. The first generation of their HIS products were developed using their own computer programming language called MIIS (Meditech Interpretive Information System) while their second generation of HIS, 1979 on, run in their own operating system called Magic. This second generation of HIS is the one known as Meditech Magic and the one that is still in use today by some health care organizations, including FHA. In the mid 80’s, MIIS was replaced by a new proprietary programming language called NPR that is still in use today in the development of their products. By the mid 90’s, they abandoned the Magic operating system and introduced their third generation of HIS base on a Client/Server architecture than runs on standard operating systems but still relies in NPR for its development. This generation of HIS is known as Meditech Client/Server. Their fourth generation of HIS, currently under development, is still based on a Client/Server architecture but is an entire redesign of all their products to make them more user friendly and keep up with the development of new technology. This last generation is known as Meditech Focus (Medical Information Technology, Inc., 2009)

In addition of using proprietary operating systems and programming languages, the first three generations of Meditech HISs relied on the use of hierarchical databases to store all the data used by the systems. This approach seems to have changed in the Focus version where the data seems to be stored in relational or object oriented databases. Hierarchical databases are rarely used nowadays; indeed, they were primarily used only in the 70’s and 80’s. Appendix A provides a brief overview about hierarchical databases.

1.2.3 Health Information Systems (HIS)

The use of Health Information Systems by hospitals and health care organizations started in the early 70’s. Since then new systems have been designed to keep up with new technological developments and the demands from their users, but direct descendants of the early generation of HISs are still in use across the country by many organizations. Most of these systems have undergone upgrades and redesigns to incorporate new features and accommodate their users’ requests but in many cases, their technology and architecture design are still tied to their origins. One example of such systems is Meditech Magic. Even though Meditech has not sold Magic HIS to their
new customers for a long time, Meditech is still supporting it and continue to improve it for their set of clients that are still using it. These clients include at least two of British Columbia health care authorities: Fraser Health Authority (FHA) and Interior Health Authority (IHA).

The use of outdated HISs provides some benefits but also some challenges to organizations like FHA. From the point of view of its users, outdated HIS provide a familiar and stable interface that health care workers have became habituated and attached to work with. On the other hand, these systems were built with technologies that in most cases are not in use today and in many cases make access to the data stored on them difficult to retrieve and share. This is in particular the case with Meditech Magic and even Meditech Client/Server. These products by themselves do not allow an easy way to consolidate the information stored in them and display it in other way than by using reports. In addition, extraction of the data is limited to be done by using download reports or using custom-made interfaces developed by Meditech itself. An alternative solution was introduced a few years ago by a third party company that developed an Open Database Connectivity (ODBC) solution that, even though it is not supported by Meditech, allows the extraction of data in real time from any Meditech HIS. ODBC is a software standard supported by most database vendors that allows any application to communicate with an ODBC aware database through the use of a driver. In the case of Meditech, the third party driver provides access to the hierarchical database using standard SQL commands rather than NPR commands.
2: REQUIREMENTS

Since the main goal of the project was the implementation of a working prototype solution, the first step taken was to determine what kind of operational information FHA’s decision makers might be interested to see on a regular basis that could help them in their day-to-day work. The second step was to find out what are the existing policies and guidelines in the organization that govern the storing and manipulating of patient data.

2.1 User’s Requirements

An initial list of possible data items was compiled from data requests received over a period of two years from different groups inside the organization and from existing reports that are sent on a daily and weekly basis to decision makers across the organization. The initial list then was submitted for review to a small group of staff and managers at the Abbotsford Regional Hospital and Cancer Centre. From their feedback, a final list was compiled and it became the blueprint to determine what data needed to be extracted from the systems and what would be displayed to the users. As stated in section 1.1, our project focuses on extracting, processing, consolidating, and displaying information about the current status of the hospitals and the entire system, although the compiled list included more than just that set of data. This was due to the fact that the exercise presented an excellent opportunity to collect information that might be valuable for the organization and could be used as guideline to expand the prototype in the future. Table 1 includes the data items that are relevant to the project. The complete list is given in Appendix B.

Current Status: Updated every 15 minutes

Resource Utilization and Operational Summary

- Emergencies
  - Number of Patients
  - Number of Admitted Patients
- Inpatients
• Number of Patients by Service Groups

Bed Availability
• Availability of Beds in the Intensive Care Unit
• Per Unit Availability of Beds

Emergencies
• Number of Patients
• Number of Admitted Patients
• Number of Patients by Triage Level (CTAS) and
  o Length of Stay since Arrival in Hours
  o Chief Complaints
  o Locations
  o Age Groups

• Number of Admitted Patients by Triage Level (CTAS) and
  o Length of Stay since Admission in Hours
  o Length of Stay since Arrival in Hours
  o Current Services
  o Chief Complaints
  o Locations
  o Age Groups

Inpatients
• Number of Patients
• Number of Patients by Service Groups
• Hospital Number of Patients by Length of Stay Groups and
  o Current Services
  o Locations
  o Age Groups
• Per Unit Number of Patients by Length of Stay Groups and
  o Current Services
  o Age Groups

In addition to provide total numbers, the interface must be able to have a drill down capability in order to get a detailed list of patients for any given totals.

Table 1 - User’s Requirements

It is important to notice that the 15 minutes update interval creates a balance between the need to have current data from the Emergency Department and the need to avoid too frequent data extractions from the HISs that might have a negative impact on their performance. Potential users agreed that the frequency of the updates would be
enough; even some of them indicated that the inpatient data set could be updated in longer intervals since changes to it are less frequent. A decision was made that in order to have a proper snapshot of the hospitals and the entire system, it was best to provide updates for all the data in the set at the same intervals.

2.2 Organization’s Requirements

At the beginning of the project, a criterion was set that highlighted the conditions that the prototype would need to meet in order to be considered as an acceptable solution. One of those conditions was that the prototype was developed following the existing policies and guidelines of the organization. In order to fulfil this condition, the following criterion was established:

• Data extraction from any of the FHA systems must not adversely impact the performance of the system in any way. This is especially critical when extracting data from the two HISs since thousands of employees use the systems across the organization, and degradation in performance might negatively affect their work.

• No detailed demographic patient information must be displayed in any way that allows identifying individual patients by name. An exclusion to this guideline is due to the fact that hospital staff, specially staff working on units, knows the patients that are under their care and could identify them by linking them to their displayed locations.

• No detailed patient medical information must be included in the extraction or display of the information other than information that might be deemed beneficial by decision makers for operational purposes.

• The extraction, storage, and display of information must be in line with FHA’s security policies and guidelines, and meet British Columbia privacy laws.
3: DATA SOURCES

Due to the fact that FHA runs two separate Meditech HISs, the data has to be extracted from both systems. In addition, in order to provide calculation for the utilization of beds, it was determined that some additional data needed to be collected and extracted from other sources because the HISs did not have it or the information that they had was inaccurate or outdated.

3.1 Meditech Systems

As indicated in the Introduction section, FHA uses two Meditech HISs: Magic and Client/Server. Meditech Magic is considered an outdated system since its technology is from the late 70’s and it was also replaced by Meditech Client/Server more than a decade ago. Even though Meditech Client/Server uses more recent technology than Meditech Magic, both systems still share a common architectural design and technology. For example, both of them use hierarchical databases to store their data and both are developed using the same proprietary programming language, NPR. Their main differences are primarily in the way that the data is processed and the capabilities of their front-ends. Like most enterprise database systems, both HISs store and process data from a centralized location but in the case of Meditech Magic, the front-end is just a text base interface that lacks any real processing capabilities. In Meditech Magic, all the processing is done in the server side, in other words it uses a centralized processing model similar to the old mainframes. Meditech Client/Server, as its name indicates, uses a distributed processing model where most of the processing happens on the server side but the front-end does more than just render what the server tells it to do. Also, in Meditech Client/Server, the front-end is a graphic interface with similar look and feel to most current Windows applications.

Meditech HIS were designed and built following the modular approach to develop systems. This means that particular functionality of the system is done by a particular module that in most cases can be purchased independently provided that the
functionality of the module does not depend in the functionality of another module. In the case of FHA, their HISs have a significant amount of modules including: System Management (MIS), Admissions (ADM), Abstracting (ABS), Emergencies (EDM), Finance (BAR), Radiology (RAD), Laboratory (LAB), Order Entry (OE), Scheduling (CWS), and Electronic Medical Records (EMR or PCI) between others. In general, the modules are available in both HISs but in some cases some of the features available in Meditech Client/Server are not available in Meditech Magic due to architectural and technology restrictions. From the point of view of the database architecture, each module maintains its own set of tables and fields organized in one or more databases. If a module is not part of the system, then its set of tables and fields is not available in the entire system. In the case of FHA configurations, Meditech Magic is the only one where some of its modules are configured to support two or more databases. This was due in part to technical restrictions in the product and also as a result of merging different Meditech systems under a single one. Figure 1 shows the basic general architecture of a Meditech HIS.

![Figure 1 - Meditech Basic Architecture](image)

Access to both systems is controlled by setting user accounts in each HIS. This means that if a user has to have access to both systems, the user is given two Meditech accounts, one for each HIS. Accounts are setup with the same names in both systems so users only need to remember one name but they are still separate accounts.

Meditech HIS use the modules and databases in conjunction with user accounts and point of access to determine who has access to what data. A user logon with a
particular account in a particular computer might have access to a certain set of data but if he moves to a different computer, he might have access to a different set of data. In this particular example, the point of access makes a difference in what the user has access to. In another example, a user might have access to a module but if the module has more than one database and he only has access to one of them, then he will only have access to the data in the database to which he has been granted permission. Some modules even allow restricted access to the data inside of a particular database to increase the security of the information inside it. This is particularly the case with the Electronic Medical Record module where detailed patient medical data is stored and made only available to medical staff. In this module, access to the data is further restricted by granting specific rights to groups of users to specific pieces of data in the database.

Like any enterprise systems, Meditech HISs try to minimize the duplication of information that they store and, to some degree, allow organizations to customize the systems to the way they run their businesses. In order to achieve these two goals, a Meditech HIS relies on a set of configuration tables, known as dictionaries, to store information that is used by the system to normalize and validate data, to adjust the way the system functions, and to support the operation of the system itself. Since dictionaries are just tables that have a special functionality inside of a module or the entire system, they are normally part of a module’s database or the System Management database. As might be expected, access to change the content of the dictionaries is restricted since changing them might affect the functionality of a module and/or the entire system.

In the case of the project, the user’s data requirements determined that the data to be extracted needed to come from three modules in each HIS, with some of the modules having more than one database. Table 2 provides a list of the modules from where the data was extracted with the number of databases that needed to be accessed in each module to pull out the data. It is important to notice that in order to get the complete data set requested by users and to present the data in a friendly manner; the set of data extracted had to come not only from regular data tables but also from dictionary tables.
As mentioned in the introduction of section 3, the data needed to calculate the utilization of beds, specifically the number of beds that are open in any given day at each unit, exists to some degree in both Meditech HISs, but after extensive examination it was determined that the data was inaccurate or outdated. The reason for this is that it is almost impossible for the staff that manage the systems to keep up with all the updates and changes that might be required in order for both systems to have accurate information at any given time. It order to reduce the amount of work, sometimes alternative solutions are put in place. This is the case with this particular set of data, where the configuration of the systems is done in such away that the systems always keep a greater number of beds available than the number of physical beds available at each hospital! This approach allows the staff using the systems to never have problems assigning beds in the systems since there are always more beds configured in the systems than the physical ones. An undesirable side effect of this solution is that it is impossible to get an accurate count of the number of physical beds available from the systems.

Using this inaccurate data was considered at the beginning of the project since it was the only one available, but during the research phase of the project an alternative source became available. As part of a FHA effort to improve the generation of a particular report that provides overall information about the daily utilization of resources in the organization, a small solution was implemented to collect the number of beds that are open on a daily basis. The solution relies on electronic forms that are filled everyday.
by the staff in charge of allocating beds at each hospital indicating the number of physical beds open at their sites. The submitted data is then collected in a database that is accessed by the person generating the report. The electronic forms used to submit the data are implemented using Microsoft InfoPath 2003. This application is a standard FHA application that is extensively used by some departments since it allows the creation of electronic forms by end users without any programming knowledge and without the need to setup any special software or application to support them.

The data submitted by the staff at each hospital is used in our project. Since the data is stored in the same relational database that is used to store the data from the HISs there was no need to develop any programs to retrieve the data from any other system.
4: RESEARCH

4.1 Data Extraction

As mentioned in section 1.2.3, until recently the only two methods to extract data from a Meditech HIS were by using Meditech’s custom-made interfaces or by creating download reports. However, a few years ago a third party company, Interface People LP (IPeople), developed an ODBC solution that allows the extraction of data in real time from any Meditech product. It is important to notice that the IPeople’s solution is not supported by Meditech.

4.1.1 Meditech Custom-Made Interfaces

Since part of the objective of the project was to come up with a solution that could be implemented with the resources available in the organization, the Meditech custom-made interface was not explored as possible part of a solution. In addition to take time to develop, Meditech custom-made interfaces are custom built to very specific requirements and lack the flexibility to handle changes. Any modifications to the interface needs to be done by Meditech in their own timeline and at a cost.

4.1.2 Meditech Download Reports

Development of download reports is one of the most used solutions to extract data from Meditech products and it is the only other solution available that is fully supported by Meditech since the development of download reports is a feature that is available in their products.

The development of reports is done in Meditech’s own Report Writer module using a subset of NPR; their own proprietary programming language that was first introduced in 1985 (Medical Information Technology, Inc., 2009). NPR is a very powerful programming language that is primarily used to create routines to perform tasks inside the HIS. Examples of these tasks include the manipulation of the data in the hierarchical databases and the control of the way information is displayed in the user interfaces. As it might be expected, NPR programs only run inside the Meditech environment, which,
from a programming point of view, could be seen as a virtual machine. There is very little information available about NPR itself, other than in Meditech user and training manuals developed to train their clients in how to use NPR to write reports and create basic screens known as Customer-Defined Screens (CDS). Meditech does not provide any development environment other than to develop reports and CDSs. Since the development of reports requires at least basic knowledge of NPR, their development is normally restricted to certain group of people inside of an organization or is done by third party companies that provide such a service. It is important to point out that running reports can have a significant impact in the performance of the system even if they are properly developed. FHA has a series of guidelines indicating when and how reports must be run. The guidelines were especially created to address download reports since these create the most overhead due to the amount of data that they might end up extracting from the HIS.

In order to download data using reports, a report first needs to be developed to extract the data from the proper module(s) using a set of parameters. In most cases, a report has access to all the data contained in any module in the system, thus the report is a very powerful tool to extract data stored in different modules. However, at the same time, cross module reports are complex, and as a result they might take longer to run and/or increase the demand of resources that in turn might end up impacting the performance of a system. Detailed explanation of how to develop a Meditech report is outside of the scope of this project; information can be found in Meditech user and training manuals available only to their clients or in training manuals developed by third party training companies like Iatric Systems Inc.

After a report has been developed and properly formatted to download data, then the report just needs to be run every time that data is required to be extracted from a system. Like in most systems, this can be accomplished in two ways: manually or automatically. In the manual method, a user manually runs the report every time that the data is needed. This is normally done when the extraction of the data is infrequent or needs to be done only once. In the automatic method, a report is scheduled to run at particular intervals of times using the scheduling functionality available on the system.

One of the advantages of using reports is that any piece of data can easily be extracted from the system since a report always has access to it regardless of its
storage location in the system. This was one of the main reasons why they were initially considered as part of the solution since the data that was needed for the project came from three different modules. The main problem with this approach is that data can only be downloaded into text files. Using download reports for data extraction would have demanded scheduling the downloads on intervals of 15 minutes to a particular network drive, and then another application would have to read the files and upload the data to a relational database for further processing/conversion. This would have demanded that the application uploading the data to the relational database would have to be synchronized in some way with the scheduling of the reports. Based on previous experience, this would be difficult to accomplish since the running times of the reports varies during the day depending on the load of the system and the amount of the data downloaded. The only way to ensure that such a solution could work would be to develop the upload application to poll the file download location to find out if a file is ready to be uploaded. The difficult part with this approach is to determine when the download has finished and if the file contains complete data or not. Due to the way that a Meditech HIS does downloads, the presence of the download file in the network drive does not indicate that the download has finished, only that it has started. In general, depending on the nature of the downloads, it could take several minutes to hours between the time that the download file was created on the network drive and the time it was finished. In the case of the data for the project, it was determined that in most cases the download would only take a couple of minutes to run making the implementation of this approach feasible in the case it was needed.

The second reason to look to the download reports as part of a solution was that this is a Meditech supported way to extract data from the systems so will always be available. Third party solutions, like the IPeople one discussed in the next subsection, are unsupported by Meditech and as a result, they might not work if Meditech makes changes to their systems in such a way that would prevent third party applications from accessing the data stored inside them.

4.1.3 IPeople Connect

IPeople is a Texas company founded by a group of former Meditech employees. IPeople specializes in creating applications that allow access to Meditech HIS data without relying in Meditech interfaces or tools to do so. The company offers many
products with different features, but there is a particular one, IPeople Connect, that provides the most functionality and flexibility. It not only allows extraction of data from Meditech databases but also submission to them. This particular product allows access to Meditech databases through the use of an ODBC driver. As shown in Figure 2, IPeople Connect is implemented as a 3-tier application where the application acts as a broker by extracting/submitting the data requested by the ODBC driver installed in the user’s computer from/to the Meditech databases.

**Figure 2 - IPeople Connect Architecture**

An interesting feature of the product is that it exposes all the Meditech hierarchical databases as relational databases. In order to extract/submit data the users just need to do so by running a standard SQL query in an application that has access to the proper ODBC driver. Also by exposing all the databases as a single relational model database, queries that extract data from different modules are easier to build than doing so in NPR download reports.

In order for a user to be able to submit a request for data to IPeople Connect, the following requirements must be met: (1) the user must have the IPeople Connect ODBC driver installed on his computer, (2) the user’s Windows account must have been granted permissions to submit requests to IPeople Connect, and (3) the user’s Meditech account must have the proper permissions in the proper Meditech HIS to extract the data he is requesting. IPeople Connect ultimately relies on the Meditech HIS security schema to control access to the data allowing in this way the group managing the HIS to keep control of what each user has access to.

IPeople Connect became available as a supported FHA application in March of 2009. FHA only uses IPeople Connect to extract data; submissions of data are not allowed since it could create serious issues in the HIS if they are not properly done.
From the development point of view, IPeople Connect provides easier access to Meditech data and it has a greater flexibility than download reports.

From the testing that was done it was clear that even when the data was exposed as a relational mode to the users, users still need to be familiar with the hierarchical structure of the databases in order to find the required data and to avoid creating queries that negatively impact the performance of the systems. This last issue is a critical issue when extracting data using IPeople Connect. Queries have to be carefully done to always use Meditech index tables as well as to avoid the use of outer join queries in any data extraction. Not following either of the above approaches results in queries that could have a very long running time and also could impact not only the performance of the Meditech HIS but also the performance of IPeople Connect.

Even with the restrictions just mentioned, the extra programming work that it would demand to address them, and the issue of using an unsupported Meditech product to extract data from Meditech systems, the IPeople Connect solution was chosen for the extraction of data for the following reasons:

- Faster and more reliable access to the HIS data since it does not require the intermediate step of downloading to a file and then uploading the information from the file to a database. This extra step would not only adds time to the overall data extraction process but also adds one more point of failure in the case that the network drive becomes unavailable.
- Stronger security since the data does not required to be stored in an intermediate file. As described in the next section, the data is just transferred directly from one database to another.

### 4.2 Data Load and Storage

From the initial conception of the project, it was clear that in order to consolidate and manipulate the data extracted from the HISs, a database has to be used. Also, since the data extracted from the HIS contains detailed patient information, access to it has to be restricted and properly secure.
The use of enterprise level databases other than as a part of a system is restricted at FHA, but access to a Microsoft SQL Server 2005 database containing similar data was available given that the proper authorization for its use was granted for the duration of the implementation of the project. An alternative option also existed, using Microsoft Access 2003, but since the extraction of the data for this project was deemed of having a potential use for the organization, authorization was given to incorporate it in the Microsoft SQL Server 2005 database. Indeed most of the data extracted for the project is now been used by the organization in one of their daily reports.

The ability to use an enterprise level database significantly simplified the implementation of the prototype since such a database provides functionality and security mechanisms that are not available in Microsoft Access 2003 and would have to be developed in some way to properly manipulate and secure the data if the latter would have been used. Even though Microsoft SQL Server 2005 was used for the implementation, only basic features available on the product were used in the implementation because FHA does not support the use of advanced features like “Reporting Services” in-house developed databases. Advanced features are only supported when they are part of application developed and maintained by external vendors. The basic features available for the implementation of any internal solution are: Primary Keys, Foreign Keys, Stored Procedures, Triggers, and the use of Schemas to control access to the data. Access to most of the functionality available in “Integration Services” is also supported but was not used in the project due to technical issues regarding the schedule of the “Integration Services” programs.

The extraction of the data from the Meditech HISs using iPeople Connect and its upload into the SQL Server 2005 database required the development of custom-made programs. As mentioned in the introduction section, development of software using professional programming tools is restricted so any development that would need to be done for the project had to be done with what it was available in all FHA computers. The standard FHA set of applications that allow running custom-made programs created using available development tools installed in all FHA computers are: Microsoft Office 2003, Internet Explorer 6.0, Adobe Flash 9.0, and Windows Scripting Host. Since the programs that extract and upload the data needed to be scheduled to run at particular intervals, it was decided that the best option would be to develop them using Visual
Basic Script (VBS). Programs developed with VBS run inside Internet Explorer 6.0 and Windows Scripting Host environment but when they run in the latter, they have access to the databases and network resources needed to retrieve and store the project’s data. The second option was the use of Visual Basic for Applications (VBA). Programs developed in VBA run inside of Microsoft Office 2003 applications but their access to resources is somewhat controlled by Microsoft Office security policies that could restrict what a program can do. Developing programs using JavaScript or ActionScript to run in Internet Explorer 6.0 and Adobe Flash 9.0 respectively were not considered at all since direct access to a database from these applications is not built in the programs.

4.3 Data Processing and Presentation

After determining how the data would be extracted and stored, focus was shifted to determine what would be the best way to process and present the data to decision makers. The selection of what to use to process and present the data was again totally driven by what technology and tools were available in FHA and their guidelines and policies regarding the use of data. These restrictions considerably limited the possible solutions, but finding out how to deal with these kinds of limitations were part of the project from the beginning.

4.3.1 Architecture

The best solution to process and present the data and the most used today is to use a 3-tier architecture. In this solution the data stored in a database, the backed-end tier, is extracted and transformed by a middle tier before is sent to be presented to the user by the front-end tier. Figure 3 shows the basic architecture of such a solution.

![Figure 3 - Basic 3-Tier Architecture](image-url)
The advantage of going with this approach is that only the middle tier needs access to the database while the front-end only needs to communicate with the middle tier. Having only the middle tier accessing the database limits the number of sessions maintained by the database to just one, while at the same time increases the security of the data by only granted access to the database to one entity. Without the middle tier, each front-end would need to establish a connection to the database and each front-end would have to have access to the database. The former adds to the workload of the database, while the latter increases the potential of security breaches.

One of the most used implementations of the above architecture is to use a web server in the middle tier and a browser as front-end. This would have been the optimal implementation for the presentation of the data for this project but due to FHA’s policies, access to a web server is limited to applications implemented by external vendors. Similar alternative implementations that rely on the development of a process running on a computer to act as middle tier and the development of a custom made front-end were also deemed not possible by the existing policies in FHA regarding the development of software. The only solution that was deemed possible and satisfied FHA policies was to adapt the 3-tier architecture to act as such but within the constraints of the environment.

The proposed architecture is still 3-tier but is one that lacks (1) the power and dynamic response provided by the middle-tier, and (2) the direct communication that normally would exist between the middle tier and the front-end. In addition, communication between the middle tier and the back-end is only direct at fixed intervals of time. The introduction of Extensible Markup Language (XML) files as a repository of data allows the front-end to have access to all the data it needs to present to the users without the need to get it directly from the middle tier. Even though this architecture does not provide the interactivity of the ones mentioned before, it achieves the same results than them for the particular requirements of the project. Since the data stored in the database only changes every 15 minutes, it does not matter if a processed copy of it is stored in a set of XML files as long as the XML files are refreshed every 15 minutes. The front-end will not get more or less data by communicating directly with the middle-tier than by getting it from the XML files. Definitely, this architecture shifts the load of the processing involved in presenting the data from the middle tier to the front-end but considering that most desktop computers underutilize their processing power this is not
a real concern nowadays (Sven Bachthaler, 2007). Figure 4 shows the architecture of the proposed implementation.

![Diagram of 3-Tier Data Processing and Presentation Architecture]

**Figure 4 - 3-Tier Data Processing and Presentation Architecture**

### 4.3.2 Programs

Like in the extraction and loading of the data, the proposed architecture required the development of custom-made programs to create the XML files and to process the data stored in the XML files and present to the users in a friendly manner.

Since the creation of the XML files requires almost the same kind of resource accesses that it was required to extract and load the data from the HISs to the database, VBS was selected for the development of those programs.
Processing and presenting the data to user requires that the custom-made program has the functionality to create complex Graphic User Interfaces (GUI) and display them in the environment that they are running. Windows Scripting Host lacks the functionality to display complex GUIs so VBS could not be used in the development of the front-end. The use of Microsoft Excel 2003 or Word 2003 were also considered as possible solutions since both of them allow the development of GUIs using VBA but they were abandoned in favour of the two remaining applications since VBA lacks proper functionality to parse and work with XML files. The choice between using Internet Explorer 6.0 with JavaScript or using Flash 9.0 with ActionScript was done solely on the ability of either of the applications to access XML files stored in a network drive. The security setting in Flash 9.0 does not allow a program running on it to access any files in the local or a network system. Flash 9.0 applications can only have access to files hosted in a web server. Programs running inside Internet Explorer 6.0 could have similar restrictions depending on the security setting of the browser but it was found that such settings were not in place for any program where the program source files reside in the same network location as the data files. Internet Explorer security allow this by default since it considers that if the source of the program is trusted then any file or program coming from the same source is also trusted. This particular finding made the Internet Explorer 6.0 with JavaScript the solution of choice.
5: PROTOTYPE

The design and implementation of the prototype was done in two phases, each taking the equivalent of two weeks of full time work over a period of two months. The first phase involved the creation of the back-end and middle tiers to extract the data from the sources, upload it into the database, and create the XML files. The second phase involved the creation of the front-end to read the data from the XML file, process it, and present it to the users inside a browser. A period of two weeks between the two phases was dedicated to troubleshoot problems and be sure that all the applications developed in the first phase were working as expected. A similar approach was done for the second phase where time was also set aside to be sure that the front-end was working properly before making it available to the users.

In order to be sure that the prototype would meet user’s expectations, as soon as the main functionality of the front-end was completed, it was shown to a small group of staff, managers, and directors at the Abbotsford Regional Hospital and Cancer Centre for their review. Their feedback was then used to do some changes in the way that some of the data was presented. Since not all the people in this group were part of the initial user’s requirements group, some additional requests were suggested during the meeting. They were deemed to be useful to have but due to the time constraints of the project, it was determined that they would not be included in the current development since the additions would involve extracting more data from the HISs and as a result modifying the extraction programs.

A possible user’s evaluation of the prototype was considered at the beginning of the project but due to time constraints, it was not possible to be done since it would have required that the users would have used the prototype for at least three to four weeks in order to get a proper feedback. Even though this was not done, the prototype was made available for a small period of time to most of the people who participated in the project at the Abbotsford Regional Hospital and Cancer Centre.
5.1 Design

The overall architecture of the prototype is shown in Figure 5. As explained in the previous section, most of the data for it is extracted from both Meditech HISs and uploaded into the SQL Server 2005 database by a set of scripts that constitute the back-end tier. The remaining data, the number of beds that are open on a daily basis, does not have to be extracted from any other system since it is already stored in the database as part of a regular collection of data done by a FHA applications. Right after the extraction and upload of the HIS data is finished, a script that constitutes the middle tier is run to create the XML files. The new set of XML files is then copied over the previous set making it available to the front-end application to read, process, and display when a user executes it.

![Figure 5 - Architecture of the Implemented Solution](image)

In order to minimize the frequency of the extractions and the amount of data extracted in each run, the extraction of the data is done in two different intervals of time: one every 15 minutes and the other once a day. The 15 minutes interval is used to extract any data that is deemed to change frequently, in other words patient related data.
The once a day interval is used to extract data that changes infrequently but it is still required by the implemented solution. Primarily this data comes from dictionary tables that in most cases are updated well ahead of the data being used by the HISs or their users.

Security is managed at two levels: access to the database and access to the XML files. Since the database has detailed patient information, access to that data is limited to the process running the programs that constitute the back-end and middle tiers. No user has access to the database in a direct or indirect way. Access to the XML files is limited to the people who are authorized to use the prototype. Access to the network folder containing the XML files is controlled using the available features existing in the operating system that stores the files. Furthermore, the XML files do not contain detailed patient information like complete names or dates of birth. This information is stored in the database but it is not made available to the front-end. The XML files only contain the data that needs to be processed and presented by the front-end; nothing more, nothing less.

Presentation of the data to the users is done in the form of a dashboard with drill down capability. It was agreed that this was the best way to do it due to the nature of the data and the need for having quick access to summary information as well as detailed information. As shown in Figures 6, 7, and 8; the dashboard presents the data in three different screens. The first one, default, provides a quick summary of the current status of the hospital or the system. This screen does not have drill down capability. The other two screens provide summaries and details information about current emergency and inpatient patients in the hospital or the system. Most of the data in these two screens is presented in summary tables with drill down capabilities. The drill down feature allows the user the option to (1) get a detail list of patients that are part of a particular displayed sub total or total, (2) sort the displayed list following a particular criteria, and (3) if it is desired, print it. Patients showing in the drill down lists are only identified by the initials of their first and last names and by a number that is uniquely assigned to each patient in a hospital at the time of their registration.
**Figure 6 - Dashboard: Current Summary Screen**

### Hospital

**Emergencies**
- All Patients: 25
- Admitted Patients: 7

**Inpatients**
- All Patients: 123

<table>
<thead>
<tr>
<th>Ages</th>
<th>ED</th>
<th>ICU</th>
<th>SICU</th>
<th>Total</th>
<th>End of Life</th>
<th>Rehabilitation</th>
<th>Psychiatric</th>
<th>Addiction</th>
<th>Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>13</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6-12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13-24</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25+</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Locations**

### Intensive Care Unit

- **Units**: 8
- **Available beds**: 7

<table>
<thead>
<tr>
<th>Locations</th>
<th>Pat</th>
<th>Available beds</th>
<th>Occupied beds</th>
<th>Unoccupied beds</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit 1</td>
<td>14</td>
<td>1</td>
<td>1</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>Unit 2</td>
<td>15</td>
<td>4</td>
<td>4</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>Unit 3</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>ICU</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Other</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 7 - Dashboard: Current Emergencies Screen**

### Current Emergencies

**Totals**
- All Patients: 25
- Admitted Patients: 7

<table>
<thead>
<tr>
<th>Triage</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>4</td>
</tr>
<tr>
<td>3-6</td>
<td>3</td>
</tr>
<tr>
<td>7-10</td>
<td>1</td>
</tr>
<tr>
<td>&gt;10</td>
<td>1</td>
</tr>
</tbody>
</table>

**Admitted Patients**

<table>
<thead>
<tr>
<th>Triage</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>3</td>
</tr>
<tr>
<td>3-6</td>
<td>3</td>
</tr>
<tr>
<td>7-10</td>
<td>1</td>
</tr>
<tr>
<td>&gt;10</td>
<td>1</td>
</tr>
</tbody>
</table>

---

27
5.2 Implementation

5.2.1 Extraction Programs

The programs used to extract the data were written in VBS and run as Windows Script Host packages from a computer running Windows XP SP3. Each package was scheduled to run at a particular point in time during the day using a scheduling program written also in VBS. The scheduling program was not developed exclusively for this project but rather it was already available and in use. Professional scheduling applications could not be used due to FHA policies, and the Windows scheduling application that comes with Windows XP lacked the detailed logging functionality needed to monitor the execution of the programs.

As mentioned in the previous section, the decision to use IPeople Connect to extract the data from the Meditech systems created some challenges to avoid performance problems: the SQL queries extracting the data must use at least one Meditech index table as the main table in the query and they must avoid the use of outer joins. The use of index tables in queries limits in some way the type of queries that could be created but this was not considered a major problem since Meditech provides index tables for almost all the data needed to be extracted for the project. The main problem came with the inability to use outer join queries during the data extraction. The parent
and child relation that tables maintain in a hierarchical database make the use of outer join queries almost a necessity since not always all the records that exist in a parent table have their equivalent records in a child table. Using an inner query between a parent and child table only returns records that exist in both tables. If the intention is to return all the records in the parent table regardless of whether they have or not records in the child tables then an outer join query is needed. For this project, this was usually the case. The solution to this issue was quite simple; just extract the data independently from each Meditech table or group of tables where an inner join would not create a loss of data, and upload the data into the database. This solution demanded more programming work since some data that could be extracted using just a single query from any relational database had to be extracted with many queries instead. But since this was the solution that provided the best performance it was the one implemented.

The way that the programs extract and upload the data to the database is as follows: (1) a SQL query is used to extract the desired data using IPeople Connect from a Meditech table or group of tables from a particular database, (2) the extracted data is then uploaded into a temporary table in the database using a Stored Procedure, and then (3) a Trigger is fired to check and move the data from the temporary table to its final destination. Since each query extracts a particular set of data, there is an equivalent set of tables in the database to upload and store it. The use of a temporary table allows a location where checks and data validations can be done before moving the data to their final destination for its final processing.

To ensure that the extraction and upload programs were quickly developed and easy to maintain, a modular approach was taken to develop them. It was decided that each process that extracted data and uploaded it to the database would have its own program. This approach not only allows for easy maintenance since each program is totally independent of each other so changes in one of them would not affect any of the other programs, but also it allows for the creation of a template that could be quickly and easily modified to meet the needs of each process.

In total eight programs were developed to extract the data from the two Meditech systems and store it in a similar number of tables in the database. Two of the programs extracted patient data from the Admission and Emergency modules, while the other six extracted data from dictionaries in the System Management and Emergency modules.
As mentioned before, programs that extract patient data are the only ones that are scheduled to run every 15 minutes so only two of the programs needed to be scheduled to run in that interval; the other six were scheduled to run just once at the end of the day.

A major concern during the creation of the prototype was the running time of the two programs that run every 15 minutes not only because of the amount of data that they needed to extract but also due to the fact that they were developed as scripts. It was found that on average the two programs take a little less than a minute to extract and save all the required data from the twelve hospitals. This leaves plenty of time for processing and room to increase the frequency of the extraction if deemed necessary.

5.2.2 Processing Program

As with the extraction programs, the processing program was written in VBS and it also runs as Windows Script Host package. This processing program executes just after the last extraction program has finished running and was scheduled as part of the group of programs that runs every 15 minutes.

The processing program runs as series of queries to create the set of XML files for each hospital and the entire system. Each set is composed of four files that contain data regarding emergency patients, inpatient patients, hospital/system bed capacity, and intensive care unit bed capacity. In order to ensure that all the files have complete data and the four files are available to the final users as a group, the creation of the files is done first in a temporary directory. When the creation of the four files is completed then they are copied over to their final destination using a single file system command. The average time it takes the program to create and save the 65 XML files needed by the solution is less than 10 seconds. This makes the average running of the extraction and processing time under 70 seconds.

5.2.3 GUI and Presentation Program

The presentation program is an AJAX application. Asynchronous JavaScript and XML (AJAX) is a technique employed to develop applications using web technologies like JavaScript, XML, Extensible Hypertext Markup Language (XHTML), Document Object Model (DOM), and Cascading Style Sheets (CSS). It is most used in the development of web applications but, as in the case of this project, it can be used in a
different setting. As any AJAX application, the presentation program is a blend of XHTML 1.0 pages containing JavaScript code and Cascading Style Sheets but instead of the data coming from a web server, it comes from a network drive in the file system.

The application was designed to run only in the Microsoft Internet Explorer 6.0 (IE 6.0) browser. The only reason for this selection is that IE 6.0 is the only browser available in all FHA computers and the only one supported by the organization.

Like with the implementation of the extraction programs, a modular approach was taken in creating the presentation application. Each XHTML page has its own JavaScript file attached to it to prevent changes in the code in one page to affect the functionality of the others. The only exceptions were common functions, all of them were placed in a file that is accessible by all the XHTML pages.

Since the front-end was just a prototype, not much effort was put in any graphic design other than to make the GUI to look organized and somewhat presentable. Instead, most of the effort was directed to functionality, making sure that the GUI was easy to navigate and presenting the data in a way that the users could easily access with the fewest amount of mouse clicks. Also attention was put on trying to restrict the amount of information presented in each screen when the user opens them. In order to achieve this, most tables shown in the GUI can be displayed as collapsed or expanded. When the user opens a particular page, only the most used tables are shown as expanded, the rest are shown as collapsed to avoid cluttering the screen.

Like with the extraction programs, performance was a concern since browser applications can be slow and the amount of data that needs to be parsed from the XML files could be substantial. In the case of the summary screens, when looking at just one hospital, the extraction of the information only takes a couple of seconds maximum, but when looking into the entire system it takes on average more than five seconds since the XML files contain data for all the hospitals. This was expected, but what was not expected was a performance issue that occurred when displaying the detailed data after a drilling down operation in a sub-total or total. Depending on the number of patients that needed to be displayed in the list, the time to display the data could go from less than a second to a time longer than the one it takes to process the summary screen. It is unclear at this time why this is happening. All the data that is parsed from the XML files
is put into arrays to speed up its future access and it is the data held in the arrays that is passed from the summary screen (parent window) to the detailed screen (child window) in what is a memory to memory transfer. This transfer should be faster than trying to read the data from a file into memory but that seems not to be the case. There is no explanation at this time about why this is happening other that there is something in the way that the browser handle the data that is slowing down the copy of data between the parent and child windows.

5.3 Results

Even though the prototype was not formally evaluated by any user, informal feedback provided by some of the users when the prototype was made available for a small period of time indicated that they considered the solution a useful tool that could be used on a regular basis. Even further, they agreed that having the information consolidated in one location and easily accessible makes it easier for them to look to the data more frequently and provide them with information than it would take longer to get from the HIS rather than just by a few clicks of a mouse.
6: FUTURE WORK

From the information gathered in the user’s requirements, it can be seen that the prototype can be enhanced to show a greater amount of information for a great period of time. Initially it was thought that it would be possible to incorporate more data and functionality than the one formally done but like in any project in order to meet the time line requirements, some restrictions had to be set in place. It is expected that the prototype will continue to be enhanced in the future since, as indicated in the result section, there is interest for it by some users to do so.

In addition to increase the amount of data displayed, one important issue that needs to be addressed is the performance problem when displaying the list of patients. This is an important functionality of the dashboard and something that some users might use frequently so it is quite important to find out why the issue is happening and if it is possible to resolve it. If the issue is related to the way that the browser just does the copy of information between parent and child window then the current implementation would have to be changed. Since it seems to be that reading the information from the XML files might be a faster way to get the data in the child window rather than doing the copy from the memory arrays in the parent window, then the programs could be changed to do so.

An additional change that was considered during the implementation of the project was a way to reduce the total time that it takes for the extraction programs to extract and upload the data into the database. The current time of less than 70 seconds is more than acceptable but if the additional data requested by some users is incorporated in the future, the total extraction time would increase considerably if the process continues to be done in the sequential order that it was implemented. The reason for doing it in sequential order is to ensure that right after the last extraction programs runs, the processing programs runs to creates the XML files. Running the extraction programs in parallel would result in a lower time but it would require the implementation of a mechanism that monitors when all the extraction programs finish running in order for the processing one to be executed. A drawback of this approach is
that would increase the load in the Meditech systems and the SQL Server database since several programs would be pulling data simultaneously from the systems and uploading it to the database increasing the number of its concurrent connections.
7: CONCLUSIONS

The project has demonstrated that it is possible for an organization to improve the access to information held in their systems even in outdated systems. In particular the project has shown that Fraser Health Authority could develop a solution to make the information in their two Health Information Systems more easily available to their users in a reliable and secure way and under the existing policies and guidelines of the organization. Furthermore, the project has proven that such a solution can be achieved using the computer and human resources available in the organization in a relatively short period of time and at low cost.
Appendix A: Hierarchical Databases

Hierarchical databases are rarely used nowadays; indeed most modern database textbooks only mention the hierarchical model as a reference without dedicating any sections to explain the topic. The main reason for it is that the relational model has become the standard in today’s world since it is able to handle almost any data modelling including the ones in the hierarchical model.

A hierarchical database can be seen as one or more tree structures where data items in the tree have a Parent-Child Relationship (PCR) between them. The data items are known as records and, like in the relational model, they are a collection of fields of the same or different data types. Each record in the tree, with the exception of the one at the root, can have none, one, or more child records but each child record can only have one and only one parent. In other words, the relation between a parent record and its children is 1:N. When a relationship of many to many (M:N) needs to be represented, the children records have to be duplicated. As it might be expected, this results in an increase in the size of the database and the complexity in the manipulation of the records since all the copies of a record need to be updated simultaneously. In order to deal with this issue, some hierarchical databases use pointers to a single instance of a record rather than creating multiple copies of the record. This approach is known as a Virtual Parent-Child Relationship (VPCR).

The main built-in constraints existing in the hierarchical model are highlighted in the lines below and they are enforced by the database management system (Ramez Elmasri, 1989).

1. No record except root records can exit without being related to a parent record. This has the following implications:
   a. A child record cannot be inserted unless it is linked to a parent record.
b. A child record may be deleted independently of its parent; however, deletion of a parent record results in all its child and descendent records being deleted automatically.

c. The above rules do not apply to virtual child record and virtual parent records. The rule here is that a pointer in a virtual child record must point to an existing virtual parent record. Deletion of a virtual parent record should not be allowed while pointers exist to it from virtual child records.

2. If a child record has two or more parent records from the same record type, then the child record must be duplicated once under each parent record.

3. If a child record having two or more parent records of different record types can do so only by having at most one real parent and all the other virtual parents.
Appendix B: User’s Data Requirements

The list below contains the complete list of data items compiled from the user’s requests over a period of two years and from the feedback provided by a small review group of staff, managers, and directors at the Abbotsford Regional Hospital and Cancer Centre.

Current Status: Updated every 15 minutes

Resource Utilization and Operational Summary

- **Emergencies**
  - Number of Patients
  - Number of Admitted Patients

- **Inpatients**
  - Number of Patients by Service Groups

- **Bed Availability**
  - Availability of Beds in the Intensive Care Unit
  - Per Unit Availability of Beds

**Emergencies**

- Number of Patients
- Number of Admitted Patients
- Number of Patients by Triage Level (CTAS) and
  - Length of Stay since Arrival in Hours
  - Chief Complaints
  - Locations
  - Age Groups

**Inpatients**

- Number of Patients
- Number of Patients by Service Groups
• Hospital Number of Patients by Length of Stay Groups and
  o Current Services
  o Locations
  o Age Groups
• Per Unit Number of Patients by Length of Stay Groups and
  o Current Services
  o Age Groups

In addition to provide total numbers, the interface must be able to have a drill down capability in order to get a detail list of patients for any given totals.

Today’s Summary: Updated every 3 hours, cumulative for the day

Emergencies
• Arrivals
  o Total Number of Recorder Patients
  o Total Number of Registered Patients
  o Total Number of Registered Patients by
    ▪ Arrival Mode
    ▪ Community
• Departures
  o Total Number of Seen Patients
  o Total Number of Seen Patients by
    ▪ Triage Level (CTAS)
    ▪ Departure Disposition
    ▪ Age Groups
    ▪ Total Length of Stay since Arrival in Hours
• Admissions
  o Total Number of Admitted Patients
  o Total Number of Admitted Patients by
    ▪ Triage Level (CTAS)
    ▪ Admission Service
    ▪ Age Groups
    ▪ Total Length of Stay since Arrival in Hours
    ▪ Total Length of Stay since Admission in Hours
• Discharges
  o Total Number of Discharges Patients
  o Total Number of Discharges Patients by
    ▪ Triage Level (CTAS)
    ▪ Admission Service
    ▪ Age Groups
    ▪ Total Length of Stay since Arrival in Hours
- Total Length of Stay since Admission in Hours

- Top 10 Chief Complaints

Inpatients

- Admissions
  - Total Number of Admitted Patients
  - Total Number of Admitted Patients by
    - Entry Mode
    - Admission Service
    - Admission Unit
    - Institution Transfer From
    - Hour of the Day

- Discharges
  - Total Number of Discharged Patients
  - Total Number of Admitted Patients by
    - Entry Mode
    - Admission Service
    - Admission Unit
    - Institution Transfer From
    - Hour of the Day

Outpatients

- Total Number of Visits
- Total Number of Visits by
  - Service Location
  - Hour of the Day

Last 15 Days Summary: Updated once a day

Provides the same level of information in the Today’s Summary but for each individual day in the last 15 days.
REFERENCE LIST

Works Cited


Works Consulted


