NETWORKED INTERNATIONAL CLASSIFICATION OF DISEASES, NINTH REVISION (ICD-9) CODER: A .NET APPROACH

By

BALAJI KRISHNAPRASAD

A THESIS PRESENTED TO THE GRADUATE SCHOOL OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2003
Copyright 2003

by

Balaji Krishnaprasad
Dedicated to my parents
ACKNOWLEDGMENTS

I thank my advisor Dr. Douglas Dankel for introducing me to this topic and giving me the opportunity to work on it. His encouragement helped me develop this system and make it easily extensible.

I also thank Dr. Christopher Sistrom for helping with implementation and the technical details of ICD codes. I am extremely grateful to Dr. Joseph Wilson for serving on my committee and providing comments to improve the project.

Last but not least, I thank my family members and friends for their constant encouragement and advice. They helped me to achieve my ambitions and perform to the best of my ability.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>iv</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>viii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>ix</td>
</tr>
<tr>
<td><strong>CHAPTER</strong></td>
<td></td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2 INTRODUCTION TO .NET AND THE ICD-9 SYSTEM</td>
<td>4</td>
</tr>
<tr>
<td>About .NET</td>
<td>4</td>
</tr>
<tr>
<td>Common Language Runtime</td>
<td>5</td>
</tr>
<tr>
<td>The .NET Framework Base Classes</td>
<td>6</td>
</tr>
<tr>
<td>Program and User Interface</td>
<td>7</td>
</tr>
<tr>
<td>Windows Forms</td>
<td>7</td>
</tr>
<tr>
<td>Web Forms</td>
<td>7</td>
</tr>
<tr>
<td>Console Applications</td>
<td>8</td>
</tr>
<tr>
<td>Web Services</td>
<td>8</td>
</tr>
<tr>
<td>Java versus .NET</td>
<td>8</td>
</tr>
<tr>
<td>2.1 Active Server Pages .NET (ASP.NET)</td>
<td>9</td>
</tr>
<tr>
<td>Hyper Text Markup Language</td>
<td>10</td>
</tr>
<tr>
<td>Hyper Text Markup Language</td>
<td>10</td>
</tr>
<tr>
<td>Server Controls</td>
<td>11</td>
</tr>
<tr>
<td>Server Side Controls</td>
<td>12</td>
</tr>
<tr>
<td>Validation Controls</td>
<td>13</td>
</tr>
<tr>
<td>User Controls</td>
<td>13</td>
</tr>
<tr>
<td>2.1.1 Microsoft SQL-Server and ASP.NET</td>
<td>13</td>
</tr>
<tr>
<td>2.1.2 The ICD-9 Code</td>
<td>15</td>
</tr>
<tr>
<td>3 USER GUIDE</td>
<td>19</td>
</tr>
<tr>
<td>Client Interface</td>
<td>19</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1 .NET framework namespaces</td>
<td>7</td>
</tr>
<tr>
<td>2-2 Classification of ICD-9 codes</td>
<td>15</td>
</tr>
<tr>
<td>2-3 Example of an ICD-9 code</td>
<td>16</td>
</tr>
<tr>
<td>2-4 Examples for a “V” code in ICD-9 code</td>
<td>17</td>
</tr>
<tr>
<td>3-1 Synonym code status symbols</td>
<td>24</td>
</tr>
<tr>
<td>4-1 Symbols used in ICD_OLD</td>
<td>29</td>
</tr>
<tr>
<td>4-2 Example ICD-9 code from full code set</td>
<td>33</td>
</tr>
<tr>
<td>4-3 Example for a synonym code</td>
<td>33</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>.NET Framework Architecture</td>
<td>5</td>
</tr>
<tr>
<td>2-2</td>
<td>Common Language Runtime (CLR) architecture</td>
<td>6</td>
</tr>
<tr>
<td>3-1</td>
<td>User login interface</td>
<td>20</td>
</tr>
<tr>
<td>3-2</td>
<td>User registration interface</td>
<td>20</td>
</tr>
<tr>
<td>3-3</td>
<td>Full-code set interface</td>
<td>21</td>
</tr>
<tr>
<td>3-4</td>
<td>Synonym addition panel</td>
<td>22</td>
</tr>
<tr>
<td>3-5</td>
<td>Client department code</td>
<td>23</td>
</tr>
<tr>
<td>3-6</td>
<td>Administrative console for user management</td>
<td>25</td>
</tr>
<tr>
<td>3-7</td>
<td>Inline editing of user data</td>
<td>25</td>
</tr>
<tr>
<td>3-8</td>
<td>Synonym status change interface</td>
<td>27</td>
</tr>
<tr>
<td>3-9</td>
<td>Synonym status change center</td>
<td>27</td>
</tr>
<tr>
<td>3-10</td>
<td>An example of voice import data</td>
<td>28</td>
</tr>
</tbody>
</table>
Abstract of Thesis Presented to the Graduate School of the University of Florida in Partial Fulfillment of the Requirements for the Degree of Master of Science

NETWORKED INTERNATIONAL CLASSIFICATION OF DISEASES, NINTH REVISION (ICD-9) CODER: A .NET APPROACH

By
Balaji Krishnaprasad

May 2003

Chair: Douglas D. Dankel II
Major Department: Computer and Information Science and Engineering

International Classification of Diseases, 9th Revision (ICD-9) is designed for use by doctors writing a diagnosis or by a coder preparing a claim form for insurance. The ICD is an alphanumeric code that is used to represent all diseases known to mankind. The Health Care Financing Administration (HCFA) issues guidelines for proper use of the ICD-9 codes. The ICD-9 code makes the diagnosis understandable for doctors and other service providers around the world. It also allows insurance claims to be settled faster.

The main aims of this project are as follows:

• To provide an efficient way to manage the code
• To make it easy for users to find a particular code
• To allow users to export the data in proprietary format to be used in other applications

This project has been developed using the .NET technology. Our application extends the functionality of earlier stand-alone applications.
CHAPTER 1
INTRODUCTION

The International Classification of Diseases (ICD-9) [American Medical Association 1998] is an integral part of patient record management. It acts as an index for diseases, allowing the system to refer to a unique code instead of having to describe the disease in detail. The ICD-9 Code is a three- to five-digit code. The first three digits give the general category (cholera is 001). The fourth and fifth digits give specifics (Salmonella Septicemia is 003.1 and Salmonella meningitis is 003.21).

Previously, doctors tried to use the codes they had memorized, because a comprehensive lookup facility was unavailable [Sistrom and Drane 2000]. This caused coding inefficiency and errors. Current systems allow doctors and coders to obtain these codes, but most of these systems are standalone applications. As a result, they are not scalable and maintaining concurrency among multiple copies is difficult. Only a few of the existing systems are distributed: eMD’s [eMD ICD-9] topsSearch™ and topsChart™; Yaki technologies™ eICD [eICD 2002]; and University of Kansas, Clinical Website’s ICD-9 code search [University of Kansas 1999].

The goal of this thesis was to develop an improvement to these other systems called ICDNeT (ICD-9 Coder New Technology). In ICDNeT the code is obtained from a central database (the only place the data ever need to be updated) that makes the code available to everyone immediately. Any user or group of users can have their own subset of codes.
This provides an advantage to departments spread around the globe. They can maintain a subset of codes they use frequently, for easier reference.

Another advantage to ICDNeT is that it is distributed, so any client can connect to the database and obtain the data. Concurrency is also maintained since the data reside in a single place. This provides freedom to an individual department since they only need to maintain their subset of codes and need not worry about the full code set, thereby saving storage space. The end user interface can be developed in whatever language the client wants. Hence, data independence exists, and the implementation details are abstracted from the data-storage level.

Data security is also preserved in ICDNeT. A user or a group of users can have access to part of the data and be restricted from accessing other parts, simplifying maintenance. The ICDNeT has all the advantages of distributed systems (resource sharing, security, scalability, concurrency, etc.). Users have access to the full code set only when they need a code that is not available in their subset of codes. This reduces the waiting time for the user by reducing the load on the server. If some particular code is being used frequently, it can be easily added to a user’s subset of codes.

ICDNeT is designed so that clients can have any backend for their local data storage; and so that any change to codes in the in the full code set can be easily cascaded to these databases. This makes the system database independent, the network technology independent, and most of all the platforms independent. To execute this program, the user needs only an internet connection and a browser capable of displaying graphics. The user is also given a choice of importing data into any customizable format (e.g.,
PowerScribe®). The program writes the data in a proprietary format so that it is compatible to the program being used, and the format can be altered as required.

The only restriction for this software is that the code must reside in a server capable of hosting ASP.NET pages. Once Microsoft releases the .NET framework for other platforms, it will become truly platform independent. The ASP.NET technology was chosen for development because the other applications being used were Microsoft applications (the MS–SQL Server® and the Internet Information Server® (IIS) web server running on a Windows 2000 Machine). Since all the technologies are Microsoft’s, better interoperability and close integration exists between them.

Chapter 2 examines three different background areas required in the development of this thesis. First, it discusses the details of the .NET architecture, its difference with the J2EE technology, and its advantages and disadvantages. Second, it discusses ASP.NET, how ASP.NET differs from Active Server Pages (ASP), its advantages and disadvantages over ASP. Finally, it provides the structure of the ICD-9 code and significance of each position within these codes. Chapter 3 deals with the details of ICDNeT by providing an overview of how the code works and the significance of each part of the code. Chapter 4 provides a more detailed view of how to use ICDNeT and other related issues including systems previously used, their design, and those currently available. Finally, Chapter 5 contains the conclusion and ideas for future expansion.
CHAPTER 2
INTRODUCTION TO .NET AND THE ICD-9 SYSTEM

The ICD-9 Coder has been developed using the .NET technology from Microsoft. This chapter discusses the key features of the .NET framework, ASP.NET, the difference between the J2EE and .NET technologies, and the structure of an ICD-9 Code.

About .NET

The .NET framework is an extensive collection of classes and objects that can be used to create web-services and deploy applications. The key components of the .NET framework are

- Common Language Runtime (CLR)
- The .NET framework base classes
- Program and user interface

The .NET framework is a run time environment that helps programmers write efficient code and provides features like automatic garbage collection. Some of the system functions are provided in classes that any programmer can use in his programs. The .NET framework is the result of years of research by the Microsoft Development Team. It has some of the rich features of other technologies (p-code in UCSD Pascal, Java Virtual Machine (JVM), etc.). The .NET framework abstracts Windows API from the development environment and allows seamless integration of code written in multiple languages (COBOL, FORTRAN, etc.). The architecture of the Microsoft™ .NET framework and its components are shown in Figure 2-1.
The Common Language Runtime (CLR) is the environment where .NET applications are executed after being converted to Microsoft Intermediate Language (MSIL). The goal of CLR is to provide a fast and efficient development environment for the programmers, automating some of the common tasks (garbage collection, etc.) and providing support and scalability (Figure 2-2).

The code that requires the CLR to execute at runtime is referred to as “managed code” [Conard et al. 2002]. The CLR plays an important role in making the .NET technology, language independent. Any application developed to run in this environment is called a managed application [Kothari and Datye 2002]. The CLR automatically makes the registered components available to other applications, without having to register the components each and every time. Also multiple versions of a component can be made to co-exist.

User code, written in any language, is compiled into MSIL, which is given to the CLR for execution. Thus any language can be used to develop any application in .NET. The only requirement is a compiler capable of compiling that language into MSIL.
The .NET Framework Base Classes

The .NET framework class library provides a rich collection of classes that can be used to create and deploy a wide variety of applications.

<table>
<thead>
<tr>
<th>Common Type System</th>
<th>Intermediate Language (IL) to native code compilers</th>
<th>Execution Support (Traditional runtime functions)</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Data types, etc.)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Garbage collection, stack walk, code manager</td>
<td>Class loader and memory layout</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2-2 Common Language Runtime (CLR) architecture

This layer is also responsible for providing I/O services, security etc. The language dependant syntax has been moved to this layer, thereby making it generic to use. This class library helps in component-based programming by providing a common substrate and API [Kothari and Datye 2002] that can be shared among different applications and development environments. Also any language based on the .NET framework has access to all of these classes, thus a program written in FORTRAN.NET can use the same class used by C, C#, etc. to perform calculations (E.g., the square root of a number is now replaced by the System.Math.Sqrt method in .NET, hence any language in the .NET framework can use this method to calculate the square root of a number). Also programmers can create classes of their own and can use them just like predefined .NET classes. These .NET classes reside inside namespace just like classes reside inside packages in the Java™ programming language. Some of the namespaces are given in Table 2-1.
<table>
<thead>
<tr>
<th>Namespace</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft .ComServices</td>
<td>Provides access to COM+ applications etc.</td>
</tr>
<tr>
<td>System</td>
<td>Provides the commonly used types</td>
</tr>
<tr>
<td>System.Collections</td>
<td>Provides classes and interfaces that describe various collections of objects, like arrays, hash tables, and other data structures</td>
</tr>
</tbody>
</table>

**Program and User Interface**

.NET provides ways for programmers to create user interfaces such as

1) Windows Forms,
2) Web Forms,
3) Console Applications, and
4) Web Services.

Each of these is discussed below.

**Windows Forms**

Windows forms, used to create Win32 applications, utilize the Windows Foundation Classes (WFC). This will be replacing the Microsoft Foundation Classes (MFC) and other Win32 development engines. It provides Application Programming Interface (API) for manipulating different Windows™ services. It is available in the System.WinForms namespace in the .NET framework.

**Web Forms**

Web Forms are a part of ASP.NET (the .NET version of Active Server Pages). Web Forms separate the design from the logic and utilize the Object-Oriented (OO) features of ASP.NET. These can be considered a template (i.e., the HTML layout) plus the elements present in the template (i.e., the server side controls.) The controls are server side components; hence manipulating them in real time is easier than utilizing client side controls. Since the controls are present in the server they can be used as objects, and all OO features can be utilized. Also the vast collection of classes present in the .Net
framework helps in creating complex applications that were previously hard to create using web interfaces.

**Console Applications**

This interface is used to write character based applications [Conard et al. 2000]. A console application refers to any application that runs in a command shell within the Windows environment. This like other User Interface is available for all .NET framework supported language.

**Web Services**

Web services helps developers utilize resources available from a remote entity to develop their products and, therefore, offer the best of different worlds to the user. A good example for this is the Google™ web services API, where the developer can use the API to build a customized web search engine with Google™ providing the search service. Web Services are revolutionizing the way web based applications are being built and are implemented as part of ASP.NET.

**Java versus .NET**

Though .NET is a relatively new programming concept when compared with Java, they still have some similar objectives including being Object Oriented and providing the programmer with rich set of classes to perform different operations, memory management, security, etc.

Several differences exist between Java and .NET:

1) Java is a platform independent language, whereas .NET technology is a language independent*. This means that the programmer can code his applications in the language of his choice, provided the programmer conforms to the CLR (Common

---

* .NET technology will become platform independent when Microsoft releases the .NET framework for other platforms.
Language Runtime) types. A programmer can write a single application in a variety of languages and can call the services of one program in a different language. This helps in reducing the training time required to program in a .NET environment.

2) The Visual Studio .NET is a complete IDE solution for developing .NET applications. There are IDEs for developing Java applications too but they are not as comprehensive or as user friendly as Visual Studio .NET is.

3) .NET has additional features that make interoperability with other .NET products robust and effective. Microsoft is developing .NET flavors of its Windows® operating system, SQL Server® etc; this will lead to a tighter integration and enhanced security between these products.

4) .NET has built in support for web services protocols like Simple Object Access Protocol (SOAP) and Universal Description, Discovery, and Integration (UDDI), which helps in building better web services applications.

5) .NET technology is based on the XML standard and hence provides strong support for XML based applications.

On the downside, .NET has some disadvantages that might prevent it from being the market leader for now. The c# language is similar to Java in many ways, like the programming style etc., but it has some of the drawbacks its predecessors (C, C++) had including allowing programmers to use pointers which has long been argued for its positive as well as negative uses. The error handling mechanism of c# is not as powerful and precise as that of Java. Since .NET is not platform independent it cannot be run on application servers running on UNIX flavored operating systems, which is prevalent in the current application servers market. The cost associated with the development tools can also be considered as a disadvantage for the .NET technology.

Active Server Pages .NET (ASP.NET)

Active Server Pages .NET version (ASP.NET) is the next version of the Active Server Pages (ASP) concept, which incorporates web forms and services. The difference between ASP and ASP.NET are [Liberty and Hurwitz 2002]:
• ASP.NET separates code from presentation (unlike ASP where the logic was embedded with `<% %>` tags inside the HTML code),

• ASP.NET code is compiled, unlike ASP where it is interpreted,

• ASP.NET code is event-driven, while ASP follows a linear programming model, and

• ASP.NET code can be written in any .NET supporting language.

ASP.NET applications can be created using a wide range of editors from notepad to Visual Studio.NET. To host ASP.NET applications, a platform capable of supporting the .NET framework and a web server capable of handling the ASP.NET requests are required.

ASP.NET forms are built using controls. They are the backbone of the User Interface. Controls range from normal buttons, to check box and complex items that can perform additional tasks. There are five types of web controls:

• HTML controls,
• HTML server controls,
• Web server controls,
• Validations controls, and
• User and custom controls.

All the controls listed above except the HTML controls run on the server side. They help in maintaining data persistence, help make applications browser independent, and allow manipulation in real time by coding etc. The HTML controls, on the other hand, exists on the client side.

**Hyper Text Markup Language Controls**

These are the normal controls available for developing HTML pages. They reside on the client side and are manipulated using scripting languages such as JavaScript, VBScript etc. To use any data associated with these controls, the data have to be posted to another page by means of a submit command. Also the controls cannot be manipulated
using Object oriented techniques, since they are not processed by the server but rather on the client side by the browser. HTML controls have been in use for a long time, but are being slowly replaced because of their limited functionality by server controls which have a higher functionality.

**Hyper Text Markup Language Server Controls**

HTML server controls are an extension of the HTML controls that handle real time server side processing including event handling, data binding, and validating the data etc. Though these pose an overhead on the server, the benefits far outweigh the disadvantages. The advantages of using server controls are:

1) They can be referenced in any code and their values can be dynamically set or read by using corresponding properties.

2) Server side controls maintain their state even during repeated postings unlike the normal controls, which are created from scratch each time the page is posted and received.

3) Event handling is possible. The controls can raise events (like the click event for a button) and thus error checking and data handling becomes easier.

Any HTML control can be converted to a server side control by adding the attributes runat = server and an id = somevalue (optional). The id attribute is used if the attribute value of that control will ever need modification. The runat attribute identifies the control to be on the server side.

A good example of an HTML server control is a date/time function. If a client side script is used, the date/time obtained would be the client computer’s date/time. To make the date/time value consistent among users, server side control can be used and assigned a date/time value. The value assigned will be the date/time on the server, and since all of the clients will be served through this server, the date/time will be consistent throughout the application, thereby minimizing errors for date/time sensitive applications.
HTML controls are further divided into input and container categories [Liberty and Hurwitz 2002]. The difference between these two groups is that the input controls have Name, Value, and Type attributes which can be modified inside the program. The container controls do not have these attributes and are instead accessed using InnerHtml or InnerText properties.

**Server Side Controls**

Server side controls or ASP controls (since they are preceded by an ASP: tag) are used to define the user interface in a web application [Kothari and Datye 2002]. They are the basis of web forms and help make them easier to understand and more intuitive. The web controls already available in ASP.NET are extensions of the HTML language, but they are not tightly bound by the syntax of the language. The syntax is just an abstraction of the language.

Server side controls help the developer in creating rich custom controls. A good example is the button control in ASP.NET which performs the same function as a submit button. This button hides many of the inconsistencies of the HTML syntax (e.g., different types of buttons - submit, reset, and a normal button - in HTML).

The server side controls help in creating cross browser and device applications by rendering the output in the format supported by the browser or device. For example, a web form when viewed by a normal browser would present the data in HTML, whereas the same output would be in Wireless Markup Language (WML) for Wireless Application Protocol (WAP) devices.

These controls help in maintaining the data between postbacks automatically, unlike previous models where the data were stored in cookies or sessions and retrieved.
This is achieved by using the viewstate (a hidden field) that holds the data through postbacks.

These also allow forms to use panels that can be made visible and invisible by changing their properties, thus allowing one single page to do what would normally take two to three pages. The server side controls have a rich collection of data bound components, which were previously available only for standalone applications written using languages like VB, etc. They can be configured at both the machine as well as the web application level.

**Validation Controls**

Validation controls help in validating the contents present in a server side control. This helps in creating error free applications. They perform the same job as a JavaScript function but are more powerful than the JavaScript functions. These controls also run in the server and can be used to validate data based on a range of values, the format of data entered, etc.

**User Controls**

User controls are those created by users and not available by default, they can be created in any .NET language and can be used by in .NET language to create a .NET application. These are similar to custom Dynamic Link Libraries (DLL) that is used to perform certain functions.

**Microsoft SQL-Server and ASP.NET**

The backend used in this project is Microsoft SQL-Server due to its integration with the .NET framework. The choice of SQL-Server as the backend was made considering the advantages of using it with ASP.NET. Both ASP.NET and SQL-Server
are Microsoft technologies and hence a tight integration exists between them. The main feature of SQL-Server used in this project is the stored procedures.

The stored procedures are used to increase the overall security of the application. The query input values are passed from the application to the stored procedures in the SQL-Server and the stored procedure gets the input and performs the sequence of functions and returns the result set.

The advantage of using the stored procedures is that the table details are hidden from the user. Hence even if the data is moved to a new table, the only place where a change has to be made is in the stored procedure and no application change has to be made. This helps in application maintenance. Stored procedures are pre-compiled and hence lead to faster execution time unlike the normal way where a SQL query has to be parsed and compiled each time it is submitted by the SQL-Server, which leads to poor response time for huge databases. The bandwidth and resource utilization is less due to the fact that the stored procedure resides within the SQL-Server itself [Tan 2002].

The SQL-Server database can be assessed from any ASP.NET application by a variety of means like using OLEDB classes or the SqlClient classes etc., The SqlClient classes are present in the System.Data.SqlClient namespace (It contains the SqlConnection, SqlCommand and SqlDataReader classes). The advantage of using the System.Data.SqlClient namespace is that it has classes that has been specifically designed and optimized just for the SQL-Server 7.0 and up versions. The OleDb classes use the Object Linking and Embedding (OLE) classes available from different providers to connect to a database, whereas the SQL classes communicate with the SQL-Server directly on the level of Tabular Data Stream (TDS) protocol, which is a proprietary
protocol used by SQL-Server for data communication [Walther 2002]. The TDS is an application-level protocol, TDS packets are encapsulated in packets built for the protocol stack used by the Net-Libraries (like TCP/IP Sockets etc.) [Tabular Data Stream 2002].

Errors relating to database querying like sql injection attacks (where a hacker includes commands in the input string to execute codes) can be avoided. Error handling issues are resolved more easily by moving the data manipulation part into stored procedures. Security can be enhanced by restricting access to data by giving the execute permission on the stored procedures only for a certain user or user group(s).

**The ICD-9 Code**

The ICD-9 Code is arranged in sorted order by the category of the disease, where the categorical grouping is given in Table 2.2.

<table>
<thead>
<tr>
<th>Code Range</th>
<th>Code Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>001-139</td>
<td>Infectious And Parasitic Diseases</td>
</tr>
<tr>
<td>140-239</td>
<td>Neoplasms</td>
</tr>
<tr>
<td>240-279</td>
<td>Endocrine, Nutritional And Metabolic Diseases, And Immunity Disorders</td>
</tr>
<tr>
<td>280-289</td>
<td>Diseases Of The Blood And Blood-Forming Organs</td>
</tr>
<tr>
<td>290-319</td>
<td>Mental Disorders</td>
</tr>
<tr>
<td>320-389</td>
<td>Diseases Of The Nervous System And Sense Organs</td>
</tr>
<tr>
<td>390-459</td>
<td>Diseases Of The Circulatory System</td>
</tr>
<tr>
<td>460-519</td>
<td>Diseases Of The Respiratory System</td>
</tr>
<tr>
<td>520-579</td>
<td>Diseases Of The Digestive System</td>
</tr>
<tr>
<td>580-629</td>
<td>Diseases Of The Genitourinary System</td>
</tr>
<tr>
<td>630-679</td>
<td>Complications Of Pregnancy, Childbirth, And The Puerperium</td>
</tr>
<tr>
<td>680-709</td>
<td>Diseases Of The Skin And Subcutaneous Tissue</td>
</tr>
<tr>
<td>710-739</td>
<td>Diseases Of The Musculoskeletal System And Connective Tissue</td>
</tr>
<tr>
<td>740-759</td>
<td>Congenital Anomalies</td>
</tr>
<tr>
<td>760-779</td>
<td>Certain Conditions Originating In The Perinatal Period</td>
</tr>
<tr>
<td>780-799</td>
<td>Symptoms, Signs, And Ill-Defined Conditions</td>
</tr>
<tr>
<td>800-999</td>
<td>Injury And Poisoning</td>
</tr>
<tr>
<td>E800-E999</td>
<td>Supplementary Classification Of External Causes Of Injury And Poisoning</td>
</tr>
<tr>
<td>V01-V83</td>
<td>Supplementary Classification Of Factors Influencing Health Status And Contact With Health Services</td>
</tr>
</tbody>
</table>
The codes are further classified inside the general group; each digit in an ICD-9 Code corresponds to precision. A suffix of “9” is an indication that the code is for unspecified symptoms for that disease. An example is given below (Table 2-3).

<table>
<thead>
<tr>
<th>ICD-9 Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>003.0</td>
<td>Salmonella gastroenteritis: Infection of gastrointestinal tract 12 to 24 hours after ingesting Salmonella organisms, causing cramps, diarrhea, nausea, fever.</td>
</tr>
<tr>
<td>003.1</td>
<td>Salmonella septicemia: Systemic Salmonella infection with toxins present in blood.</td>
</tr>
<tr>
<td>003.2</td>
<td>Localized salmonella infections: Salmonella infection restricted to specific organ or locality.</td>
</tr>
<tr>
<td>003.20</td>
<td>Localized salmonella infection, unspecified.</td>
</tr>
<tr>
<td>003.21</td>
<td>Salmonella meningitis: Infection of membranes of brain or spinal cord.</td>
</tr>
<tr>
<td>003.22</td>
<td>Salmonella pneumonia: Infection and inflammation of lungs.</td>
</tr>
<tr>
<td>003.23</td>
<td>Salmonella arthritis: Infection and inflammation of joints.</td>
</tr>
<tr>
<td>003.24</td>
<td>Salmonella osteomyelitis: Infection and inflammation of bone.</td>
</tr>
<tr>
<td>003.29</td>
<td>Other.</td>
</tr>
<tr>
<td>003.8</td>
<td>Other specified salmonella infections.</td>
</tr>
<tr>
<td>003.9</td>
<td>Salmonella infection, unspecified.</td>
</tr>
</tbody>
</table>

The “V” affixed codes are used only when ancillary services (e.g., physical therapy) are provided to the patient. They are used when the disease or injury are not classified under the general codes (001–999) as “diagnosis” or “problems”. This can occur due to three reasons.

- A person has to undergo health services (when not sick) for any special purpose, like organ donation etc.,
- When a person undergoes a specific treatment for a known disease that person has (E.g. dialysis for renal disease).
- When a person’s health is influenced by circumstances or events but is not an illness or injury by itself.

An example for “V” code is given in Table 2.4.
### Table 2-4 Examples for a “V” code in ICD-9 code

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>V10.5</td>
<td>Urinary organs: History of conditions classifiable to 188 and 189</td>
</tr>
<tr>
<td>V10.50</td>
<td>Urinary organs, unspecified</td>
</tr>
<tr>
<td>V10.51</td>
<td>Bladder</td>
</tr>
<tr>
<td>V10.52</td>
<td>Kidney</td>
</tr>
<tr>
<td>V10.59</td>
<td>Other</td>
</tr>
</tbody>
</table>

The “E” codes are used to classify environmental events, circumstances, and other conditions that are the cause of injury and other adverse effects [American Medical Association 1998]. The “E” code is intended for use in addition to a main code (E908.1 corresponds to injury caused due to Tornado, E906.8 corresponds to injury caused due to being trampled by animal and E828 corresponds to being trampled by animal being ridden, in sport or transport). The fourth and fifth digit is used to identify the injured person.

Some of the ICD-9 codes will not be accepted by insurance companies unless the most precise numbering for that code is present. In order to obtain a reimbursement from an insurance company a medical necessity has to be established. The medical assistance provided has to be justified by presenting the required facts. In order to access the need for care, the insurance company requires the following information:

- The severity of the patient’s injury or illness.
- The reasons describing the medical care provided to the patient.

The above has to be substantiated in the patient’s medical record. The initial code in the HCFA-1500 claim form must describe the most important reason for providing the treatment to the patient. The ICD-9 codes must be listed in the order of the importance; also the code for the treatment that was given currently to the patient should be used. The unspecified code is used only when the medical test results are not confirmed, all other
unspecified code is used only when the medical test results are not confirmed, all other times the code is used to the maximum precision level in order to expedite the claims process. The ICD-9 codes help in identifying whether the treatment given falls under the medical necessity. The precision of the ICD-9 codes helps in identifying the seriousness and the extent of the injury or illness and thereby helping to establish how much of the cost will be covered by the insurance company. Thus whether a claim will be accepted or not lies with the medical practitioner in using the correct code with the required degree of precision. Also if a particular disease or illness can be classified under different codes, the medical practitioner has to use the one that would be more appropriate in that case.
CHAPTER 3
USER GUIDE

The previous chapter discussed the technologies used in ICDNeT, the advantages of those technologies and the architecture of the ICD-9 codes. This chapter presents the interface of ICDNeT and a detailed description on using the ICDNeT system.

ICDNeT has a client and administrative interface. The client interface is divided into 2 major parts:

- The client full-code list interface and
- The client department code (synonym) interface.

There are a series of screens within these interfaces. The client part of ICDNeT has a login and register screen, the details of which are discussed later in the chapter.

**Client Interface**

**Client Full Code Set Interface**

The login and registration screens are given in Figures 3-1 and 3-2, respectively. The registration page verifies that the userid is in the database and displays the appropriate message. The login page authenticates the user and sets a session variable based on the user’s authorization status (a return code of 1 for authorized user and a return code of 2 for unauthorized user.) If the user is not authorized he is given a message detailing the reason for failure of authentication.

The session variable is active as long as the user is actively using the site, but expires if the user does not perform actions for an extended period of time or the browser window is closed. In ASP.NET, sessions are maintained as either a cookie on the client
machine or, if cookies are not enabled, by redirecting a request to a URL that has the ASP.NET session ID. When a request is received, the embedded session ID is taken from the URL and is used to find the appropriate instance of the session object [Powell 2002].

Figure 3-1 User login interface

Figure 3-2 User registration interface
Once the client has logged on the system, the interface is used to provide the full ICD-9 Code set to the user. The user can search the full-code list based on the code name or the code itself and can sort the results obtained either by the code or by the description of the ICD-9 Code. See Figure 3-3. When the user clicks on the required code it is copied to the system’s clipboard and is displayed in the Clipboard label box. The user can click on the name (or description) of the code to view the synonyms used. See Figure 3-4.

The user is allowed to access to the addition of synonym only if he/she has login rights to the system, otherwise they are only be able to view the synonym. Users needing to modify the synonym list must register first, and then they can request permission from the system administrator who grants this access on a case by case basis.

![Figure 3-3 Full-code set interface](image)

Note that the synonym panel can be customized as needed. This is performed by manipulating the display’s stored procedure. This customization can be used to obtain the data from the required table for any particular department or change the way the
information is displayed. Paging is enabled since the results returned may exceed one page. The data are maintained in the clipboard even when paging through the records.

---

**Figure 3-4 Synonym addition panel**

**Client Department Code (Synonym) Interface**

The Client Department Code List (Synonym) interface is similar to the Full-Code set; this was done to maintain a standard interface throughout the site. The major difference in the department interface is the addition of a status column that provides a visual description of the status of the ICD-9 code. Figure 3-5 details the department code
list interface and Table 3-1 gives the legend for the status symbols. The synonyms are initially assigned a status of “Ok Code” meaning that the code has not yet been reviewed so any other status can be assigned. Once the code status administrator reviews it, the status is changed from “Ok Code” to the appropriate code.

![Figure 3-5 Client department code](image)

**Administrative Interface**

The administrative interface is divided into user administration and data administration.
Table 3-1 Synonym code status symbols

<table>
<thead>
<tr>
<th>Status Code Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Red Symbol]</td>
<td>This symbol describes the code as being a “Bad Code.” Due to reasons like being rejected by Insurance companies etc., these codes must be avoided as much as possible.</td>
</tr>
<tr>
<td>![Green Symbol]</td>
<td>This symbol describes the code as being a “Good Code” that which can be used in coding patient records.</td>
</tr>
<tr>
<td>![Gray Symbol]</td>
<td>This symbol also describes the code as being a “Good Code,” as well as indicating that the code is already present in the PowerScribe® application.</td>
</tr>
<tr>
<td>![Yellow Symbol]</td>
<td>This symbol is the default symbol indicating that the code has not yet been classified as either a good or bad code.</td>
</tr>
</tbody>
</table>

User Administration

User administration is designed to give the administrator the ability to manage the account of a user from a single place of operation. The administrator(s) of ICDNeT can:

- Authorize or deny permission to add synonyms,
- Edit user name information,
- Delete a user account, and
- Reset a user password.

The reset password feature has been used to utilize password encryption technology in future releases, thereby preventing anyone from viewing the password of the user. The editing feature used is the in-line editing facility present in a datagrid (Figure 3-7). The only editable feature as of now is the user name. The interface is shown in Figure 3-6.

Note that the administrator account cannot be denied permission even if the Authorize / Deny button is pressed. This protects the administrator’s account from being locked out.

Synonym Administration

The synonym administration screen (See Figure 3-8) is used to control the synonym data. It provides of the following actions:
Figure 3-6 Administrative console for user management

Figure 3-7 Inline editing of user data

- Search for a synonym based on code number or name,
- Change the status of the synonym (Table 3-1), and
- Delete the synonym if it is not required.
The search functionality is similar to the search in the client interface. Once a code is returned, the user can click on it and change its status. The status code interface (see Figure 3-9) allows the administrator to change the status to one of the five options given in Table 3-1. If the synonym status is changed to a good code then all the synonyms with that associated number become good codes automatically. This similarity rule applies to all the other codes, except for the voice-import status. This changes the associated synonyms to good, except for those that already have the voice ready status in that code (which will be retained).

If a code is made voice ready then a flag is set. This flag identifies to the administrator that when he imports the voice codes into the PowerScribe® format, this code should be imported.

As indicated the status codes are used for information purposes only. ICDNeT will not prevent the user from using codes that are classified as bad or ok codes. Which code is used is left to the discretion of the doctor who writes the report.

The PowerScribe® import list is populated with the “voice ready” codes when the administrator uses the Voice Import menu. Figure 3-10 lists an example of the import document; it is the responsibility of the administrator to import the data into PowerScribe® on a regular basis.

The previous chapters gave an introduction to .NET and other technologies used in ICDNeT. This chapter gave an overview of the working of ICDNeT and an explanation of its interfaces. The next chapter provides details on its implementation and the advantages of ICDNeT. The final chapter provides areas for enhancement in ICDNeT.
Figure 3-8 Synonym status change interface

Figure 3-9 Synonym status change center
Figure 3-10 An example of voice import data.
CHAPTER 4
ARCHITECTURE OF ICDNeT AND COMPARISONS

The previous chapter discussed the how-to’s of ICDNeT system; this chapter
discusses the architecture and advantages of ICDNet system and how it compares with
currently available systems.

**Previous System**

The current ICD-9 Coder system, that is about to be replaced, is a standalone
version [Sistrom and Drane 2000] created using MS-Access™ database and Forms
present in MS-Access™ (ICD_OLD).

At the time it was created, it was justified as a standalone product. But with the
advent of the web and client server technology, it became pertinent to create a distributed,
scalable, and robust application. ICD_OLD maintained concurrency by making the user
manually update the database before using the system. The user could search the
department’s code list or the full list, copying the code of interest to the clipboard so it
could later be pasted in the patient’s report. This system used symbols and color code to
indicate the status of each code (given in Table 4-1 below)

<table>
<thead>
<tr>
<th>Status Long</th>
<th>Display Color</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>No status assigned</td>
<td>black</td>
<td>0</td>
</tr>
<tr>
<td>Valid, uncommonly used</td>
<td>green</td>
<td>+</td>
</tr>
<tr>
<td>Valid, commonly used</td>
<td>green</td>
<td>+</td>
</tr>
<tr>
<td>Problem code use rarely if ever</td>
<td>yellow</td>
<td>?</td>
</tr>
<tr>
<td>Problem code do not use</td>
<td>red</td>
<td>-</td>
</tr>
<tr>
<td>Need 5th digit for this code</td>
<td>yellow</td>
<td>?</td>
</tr>
<tr>
<td>Need 4th digit for this code</td>
<td>yellow</td>
<td>?</td>
</tr>
</tbody>
</table>
It was impossible to keep a user informed of the changes to the database without the user updating his local copy from the main table on a network drive, this posed some potentially serious problems such as if the network path is renamed or the file is moved from one place to another the user would not know where to find the database.

Security is also a key problem in standalone systems where there are multiple points of failure. The system can be compromised if any user’s account is hacked or the user leaves his computer without locking or logging off and another person uses the system. The malicious user can delete or modify codes in the main table, thus making way for serious consequences due to the fact that human lives are involved, this is not the case with client-server applications where there is only a single point of failure (i.e., the security of the server being compromised) which can be prevented by applying the necessary patches and auditing the security logs for intrusions and preventing unauthorized users from accessing the system.

Another problem with stand-alone applications is that any change to a system requires the distribution of the entire application to every person. Problems will exist for a user using a previous version such as losing some added functionalities or, more importantly, using a version with bugs. This was one of the prime reasons to move to a distributed application.

**ICDNeT Architecture**

The ICDNeT was developed as a distributed application using ASP.NET (specifically VB.NET) and the backend database MS-SQLServer®. The whole application is designed to run on any Windows machine running any Internet Information
Server® (IIS) version capable of supporting the .NET framework and the .NET framework. The application is restricted to the Windows platform only because the .NET framework is not currently available for other platforms.

The client and administrator interface has been implemented entirely using ASP.NET. Chapter 2 discusses the advantages of using ASP.NET as the programming tool. This chapter discusses the particular objects used in the ICDNeT implementation and their individual advantages.

The users are given the choice of logging in or just browsing the code without logging in to the system. The client screen uses the Textbox object to obtain data from the user, and a button control to submit the entered data. The Textbox (Server side control) was used instead of the HTML Server control (i.e., a normal text box with the runat attribute set to server) to make the rendering of the Textbox device independent.

The input data are then given as the input parameter to a stored procedure running on the SQL-Server. The stored procedure is pre-compiled by the SQL-server and executes the query on the database returning the results to the application.

The datagrid control [Datagrid Class Information 2001] is used to display the data to the client. The datagrid control is a versatile control available in ASP.NET, which has many built-in features that can be manipulated to get the desired output. One of the many advantages of using a datagrid is that it displays the data in a tabular format automatically. The markup output for a datagrid is that of a <TABLE> element in HTML or its corresponding element in other markup languages.

Another advantage of using a datagrid is the in-place editing feature whereby the data in a datagrid row can be edited on the datagrid itself without having to open a new
page. It is done using a special column called *EditCommandColumn* [Esposito 2002] that must be present in the datagrid (one per datagrid) to activate the in-place editing feature. But, the update code must be written by the user to indicating which table in the database will be updated. The appearance of the datagrid can be controlled by setting the appropriate value in the datagrid properties. Templates can be used to display required data, this helps in customizing the output to suit different user(s) or group(s).

Datagrid has additional features like caching of the data, which helps in reducing the retrieval time for the data. The data are cached on the server and each subsequent hit is read from the cache, which is faster than reading from a datasource like a database. This property of datagrid has been used in ICDNet since the ICD-9 Codes are not frequently deleted or inserted (except in the department code list). By caching the data, the number of calls to the database is reduced thereby reducing traffic between the web server and the database server. Caching also has other advantages like serving a larger number of users or as many users the web server can support since the data is coming from the cache rather than the database server which could have a limited number of database connection threads.

In the complete list of ICD-9 Codes table there is a status field indicating the level of code: whether it is a top-level code (in the hierarchy), a fourth digit code, or fifth digit code. This has been done to make the code’s status more visually available.

The user has the choice of copying the code onto the clipboard to use in other applications and to view any synonyms associated with the code. Synonyms are the names given to a particular code by doctors to make them easily to remember. For
example, the code for CNS Tuberculosis is given in Table 4-2, and its corresponding synonym is given in Table 4-3.

Table 4-2 Example ICD-9 code from full code set

<table>
<thead>
<tr>
<th>Code</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>013.</td>
<td>Cns Tuberculosis*</td>
</tr>
<tr>
<td>013.0</td>
<td>Tuberculous Meningitis*</td>
</tr>
<tr>
<td>013.00</td>
<td>TB Meningitis-Unspec</td>
</tr>
<tr>
<td>013.01</td>
<td>TB Meningitis-No Exam</td>
</tr>
<tr>
<td>013.02</td>
<td>TB Meningitis-Exam Unkn</td>
</tr>
<tr>
<td>013.03</td>
<td>TB Meningitis-Micro Dx</td>
</tr>
<tr>
<td>013.04</td>
<td>TB Meningitis-Cult Dx</td>
</tr>
<tr>
<td>013.05</td>
<td>TB Meningitis-Histo Dx</td>
</tr>
<tr>
<td>013.06</td>
<td>TB Meningitis-Oth Test</td>
</tr>
</tbody>
</table>

Table 4-3 Example for a synonym code.

<table>
<thead>
<tr>
<th>Code</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>013.0</td>
<td>Meningitis: tuberculous</td>
</tr>
<tr>
<td>013.0</td>
<td>Tuberculosis, meningitis</td>
</tr>
<tr>
<td>013.2</td>
<td>Tuberculoma brain</td>
</tr>
<tr>
<td>013.3</td>
<td>Abscess, tuberculous</td>
</tr>
<tr>
<td>013.3</td>
<td>Tuberculosis, brain abscess</td>
</tr>
<tr>
<td>013.52</td>
<td>Spinal tuberculosis</td>
</tr>
<tr>
<td>013.60</td>
<td>Tuberculosis of CNS</td>
</tr>
<tr>
<td>013.9</td>
<td>Tuberculosis, unspecified</td>
</tr>
</tbody>
</table>

An authorized person can add a synonym to the department code list by logging in to the system. Once the user is authenticated his login information is stored in a session object. The advantages of using a session object over a cookie is that for cookies to be used the cookie feature must be activated in the internet browser properties. In addition, managing sessions are easier and faster than processing cookies. Sessions are stored on the server side whereas the cookies are stored on the client. And session objects are destroyed when the browser is closed, or can be deleted after a predefined time, thereby increasing application security.
In ICDNeT, security has been implemented by utilizing session variables. If the session variable is not set then the person can only browse the code and look at the synonyms. The section allowing the addition of synonyms is protected by being enclosed within a panel object, which has its default setting to invisible. It is made visible only if the session variable is set properly. Since the panel resides within the same page, it is almost impossible to access the data unless the panel’s visible property is set to true.

The administrator interface has a similar layout with the added functionalities of authorizing a user account, denying access to a user account, adding or removing a code from the full code set or the department code list, changing the status of the code (Bad, Good, Ok, etc.), and downloading selected codes to be imported into other applications (e.g., PowerScribe®).

The page layout has been designed to give contrasting colors that help in distinguishing one row of data from another, but at the same time not being of such a high contrast to distract the user from the main purpose of retrieving the code. The colors have been chosen so that whether the room is brightly lit or dark, the code and its description will stand out.

Other Systems

There are many web applications that help in ICD-9 browsing which are available, but each has some limitations that has been identified and eliminated from ICDNeT. Each of these systems are discussed and compared with ICDNeT in the following pages of this chapter.

e-MDs ICD-9 Search

The e-MDs ICD-9 Search (EMD) is part of their topSearch® and topsChart® suite. The similarities between EMD and ICDNet are:
• Colorizing of the different codes to visually distinguish between a code that requires a fourth or fifth digit, etc.,
• Restricting the output data based on the search criteria entered, and
• Displaying as many data as there are present and not restricting the size of the output.

The EMD is restricted in many ways:
• The version available on the web is a trial version and the whole search engine is part of the bigger suites (topSearch® and topsChart®) available for purchase,
• The “E” and “V” codes appear to not be available in this version,
• The user cannot copy the code to the clipboard for use in other applications, and
• It does not allow importing ICD-9 code data.

The EMD solution is effective if their system (topSearch®, topsChart®, etc.) is used for billing and other functions. But the cost benefit of EMD over ICDNet makes it less useful in places where only the code is required, and additional functions to make the code available to other applications are necessary.

Yaki Technologies EICD (eICD)

The eICD has been designed and developed by Yaki Technologies. The similarities between ICDNeT and eICD are:
• Both have facilities to search by text present in the code,
• Both are distributed web applications, and
• Both do not limit the number of codes returned.

The differences between ICDNeT and eICD are:
• ICDNeT allows the user to search based on the code whereas eICD requires the user to click on the link that corresponds to the group to which the code belongs; this is restrictive in that the user must scroll through a whole list of code where he might need one of those codes,
• eICD does not have the feature to copy the code to clipboard for use in other applications,
• eICD does not allow importing of required data, and
• eICD does not have visual indications for fourth and fifth digit requirements.

University of Kansas ICD Browser (UKICD)

The University of Kansas Medical Center maintains an ICD-9 browser. It is similar to ICDNeT in some aspects like being able to search based on the code or number. Some of the advantages of the UKICD are that it allows people to perform complex queries like searches involving multiple codes, but this feature is not a necessary feature due to the fact that doctors mostly search based on the code name. Some of the disadvantages of the UKICD are:

• No provision to copy data to clipboard to use in other applications,
• No provision to import data in the format required,
• Number of results returned is restricted to one hundred codes, and
• Does not have visual coding assistance features (like indication of good code or bad code, etc.).

Of the three systems discussed above the eICD and EMD require registering to use the full functionality of the respective systems. eICD is a free product but requires registration by the medical practitioner who would like to use it. The advantages of ICDNeT are discussed in detail in the reminder of this chapter.

Advantages of ICDNeT

The ICDNeT has been designed to overcome the problems of the previous systems and to exceed the advantages of available systems.

Each row of data is color coded in contrasting colors to make it more readable to the user. The color coding helps in identifying the data more easily. In addition to using color coding for the data rows, symbols are used to indicate the status of the code being
examined. The symbols are both color coded and geometrically designed, like a red hexagon to indicate a bad code, a green rounded rectangle to indicate a good code, a yellow circle to indicate an ok code (which is assigned initially before any status is assigned to the code), and a microphone symbol to indicate that the specific code is available in PowerScribe®. These codes can be customized to suit the requirements of the department using the system.

The ICDNeT allows the user to copy data onto the clipboard so that multiple codes can be copied and pasted into the patient’s record if that particular symptom or injury requires classification with multiple codes.

The ICDNeT allows the code manager to modify the status of the code in the department list based on the feedback obtained from insurance claims.

The ICDNeT allows the user to sort the codes based on either the code name, code status, or the code itself in an ad-hoc fashion. Also the user can browse through multiple pages and can add a synonym to any code. The ICDNeT performance is one of its major advantages, since the page is cached, the sorting and pagination through the data is faster than it would be to send a query to a database and receive the results.

The ICDNeT allows users to take advantage of the central code repository; hence any developer who wants to use the full-code set can make use of the services being offered by ICDNeT. The user must register to obtain an authentication id and password. They will have full access to the codes, but will be restricted to only viewing the data and not being able to manipulate it. Whenever the data are updated, they will be available for all the users connected to the database.
For example, if medical practitioner running his own clinic uses only a few subsets of codes, he can either browse for the code through the web or use a small footprint application that would store his subset of data, updating it from the full list whenever the main table is updated.

The importing facility present in the ICDNeT can be customized to present the data in any required format, specified by the administrator account. In the Radiology department, the import function presents the data in a format suitable for PowerScribe®, which is a voice recognition patient management system. In this system the doctors dictate the patient’s report and add ICD-9 codes to indicate the disease. The codes that are frequently being used can be assigned a special code status (identified by a microphone symbol). These codes are imported on a frequent basis to the PowerScribe® ICD-9 content by the system administrator, thereby allowing the doctors to dictate entire reports without having to change applications to obtain the required code.

ICDNeT was designed to be backward compatible with the previous standalone version, by using the same database fields. This provides users a seamless migration from the standalone application to the web interface. The interface was design to resemble the interface of the standalone application, easing the burden of the previous system’s users.

**Installation Requirements for ICDNeT**

The ICDNeT is designed to run on a Windows™ platform, since the .NET framework is available only for the Windows™ platform so far. Because it is a web application, it requires a web server capable of handling the load as well as the ability to server ASP.NET pages. In the current implementation, Internet Information Server (IIS) version 5.0 has been used. A database to hold the data is required in the backend.
Microsoft SQL-Server has been used as the backend in the system implemented in the Radiology department of Shands at UF.

The requirements listed above are for the server side implementation, the requirements on the client side are; a browser capable of displaying ASP.NET pages. Most of the modern browsers (e.g., IE versions 5.0 and up, and Netscape versions 6.0 and up) are capable of displaying the ASP.NET page content. JavaScript must also be enabled on the client to execute some of the scripts generated.

**User Interface Design**

The User Interface (UI) for ICDNeT follows the general internet standards designed for clear and legible delivery of the content to the user. The content has been so designed to fit inside an 800 x 600 screen resolution (which is the default in most places). According to a research conducted by MyComputer.com in January of 2001 around 50% of the users had a screen resolution of 800 x 600 [Bernard 2002]. Although ICDNeT has been designed for smaller resolution, tests conducted on higher resolution monitors showed good display performance. The font used in the display is Ariel 12 point; which has been proved to be the most legible font according to research surveys [Bernard 2002].

ICDNeT makes use of layers instead of frames to present data in a page, since frames have always proved to be problematic [Nielsen 1996]. If the search results were presented inside a frame, the link in the search result page would point to the frame page thereby placing the user in a page without any other details such as a menu frame, etc. Layers help remove this drawback and are easy to manipulate.

A layer can be placed in any part of the page and multiple levels of layers can co-exist. Through scripting we can manipulate the layers to hide one layer while making another visible.
ICDNeT uses the library feature of Macromedia Dreamweaver MX™ for repeating elements like the banner and the menu layer. The advantage of using a library is that modification needs to be made in the single location with the changes reflected across all the pages having a reference to the library. This is particularly useful in the scenario described above because the banner is an element which needs to be present throughout the website. By using the library for the menu bar we need not worry about broken links since items are replaced in the library file alone and are updated everywhere.

The details of ICDNeT, its features and comparison with other systems have been discussed in this chapter. The final chapter highlights the future extensions possible to ICDNeT and concluding remarks.
CHAPTER 5
CONCLUSION AND FUTURE EXPANSION

The ICDNeT has been developed to be a one-stop solution for all needs of a patient management system in disease coding. The many advantages of ICDNeT (like being able to copy the code on to the clipboard for use by other applications, ability to lookup the synonym of a code or vice versa, ability to add synonym and import codes in proprietary format etc..,) and its disadvantages (like being bound to the Windows Platform etc..,) have been discussed in the previous chapters. ICDNeT has several areas available for future expansions like enhanced security, supporting a wide variety of users based on customizations, etc. All of the above can be achieved without having to alter the basic functions provided by the base product. This chapter provides an insight into the future expansions that can be made to the ICDNeT system in order to further improve its usability and performance.

Security

Security is a constant issue of concern in the computer industry. ICDNeT uses the .NET security model in addition to the NTFS (NT File System) already present in the system. This method, though not foolproof, is enough to stop the majority of the hacking community from getting into the system. By using stored procedures the security of the SQL-Server is maintained. Future versions of ICDNeT can be made to integrate with the Active Directory® environment (if present) of a system.
Active Directory is a component of the Windows™ 2000 environment. It helps in managing users and computers through a central command unit [Microsoft Active Directory 2000]. It provides directory services for distributed computing environment. Active Directory helps in securing a system by implementing group policy (one of the key features of the Active Directory service), which helps in assigning rights to group of users instead of individual users. By integrating ICDNeT with Active Directory the user need not login to the system to use the features that require a login, instead their credentials will be obtained from their domain login and their privileges will be assigned automatically. Thus the end user will just browse the site through the Active Directory domain to be authenticated. This can also be further enhanced by automatically adding the name of the user in the medical report, thereby customizing the entire process based on the user.

**Web Services**

The ICDNeT can also be converted to a web service that allows people to plug into the ICDNeT system, obtain the data they require, and then modify and display them accordingly. A web service is a method of integrating different services offered by content providers to be used in varied applications. A good example is the Google™ web services. Using the APIs provided, a developer can customize the results of the Google™ search engine. By using the web services a vendor can expose components to users who can use it to customize the results in their own way. Some of the key technologies driving web services are SOAP (Simple Object Access Protocol) and XML (eXtensible Markup Language) [Tidwell 2000]
The current feature available in the ICDNeT allows users to customize their UI by just giving the data through a stored procedure to which a person or a group of people can be given access.

**Future Enhancements**

Using WAP technology and WML, ICDNeT can be made available through wireless voice enabled devices (e.g., cellular phones). In this manner a medical practitioner can dial a number and an automated response system can obtain the ICD code required.

The User administration can be further enhanced by adding facilities for including details of the user and the most frequent codes used by the user to give a personalized effect.

The import functionality can be enhanced by making it automatic instead of the administrator manually importing the voice codes into the PowerScribe® system. Current version of PowerScribe® does not allow for modification of its proprietary database where it stores these data. It is anticipated to be available in the future versions of PowerScribe® and the functionality can be added into the ICDNeT system without having to modify the base system.
APPENDIX A
TERMINOLOGIES USED

Some of the key terminologies used throughout this thesis will be explained briefly here.

**ICD**: ICD refers to International Classification of Diseases and is used to refer to diseases by code number.

**ICD-9**: ICD codes 9th revision as issued by the Health Care and Financing Administration (HCFA).

**ASP**: Active Server Pages, a technology developed by Microsoft™ for building web applications.

**.NET**: “A prefabricated infrastructure for solving common problems in internet applications.” [Platt 2001]

**ASP.NET**: The .NET version of Active Server Pages technology (ASP).

**SQL-Server**: Industrial strength database product from Microsoft™.

**PowerScribe®**: Dictaphone™ PowerScribe® is an integrated speech recognition, dictation and medical transcription management system developed specifically to reduce the high cost of radiology reporting.

**Datagrid**: Datagrid is ASP.NET server control, and is used to display data dynamically. It can be bound to a datasource and the resulting output can be formatted in the required fashion.
APPENDIX B
MODIFICATIONS AND REVISIONS FOR ICD-9

The Health Care and Financing Administration (HCFA) issued ICD-9-CM (Clinical Modifications) in July of 1998 and went into effect the October 1st of 1998. 65 new codes were added, 12 codes were invalidated and 6 code descriptors were revised. The changes made are listed below.

**Diseases of the Respiratory System**

Code 482.4, "Pneumonia due to Staphylococcus," is now invalid with the addition of three new five-digit codes

- 482.40—Pneumonia due to Staphylococcus, unspecified
- 482.41—Pneumonia due to Staphylococcus aureus
- 482.49—Other Staphylococcus pneumonia

Two new five-digit codes were added under the "Other diseases of lung" heading

- 518.83—Chronic respiratory failure
- 518.84—Acute and chronic respiratory failure

The descriptor for code 518.81, "Respiratory failure," was revised to “Acute respiratory failure”.

**Diseases of the Digestive System**

Four new five-digit codes were added under the "Disorder of function of stomach" heading

- 536.40—Unspecified gastrostomy complication
- 536.41—Infection of gastrostomy
- 536.42—Mechanical complication of gastrostomy
- 536.49—Other gastrostomy complication
Code 564.8, "Other specified functional disorder of intestine," is now invalid with the addition of two new five-digit codes

- 564.81—Neurogenic bowel
- 564.89—Other functional disorder of intestine

Symptoms, Signs and Ill-defined Conditions

Code 780.7, "Malaise and fatigue," is now invalid with the addition of two five-digit codes

- 780.71—Chronic fatigue syndrome
- 780.79—Other malaise and fatigue

Five new five-digit codes were added under the "Dyspnea and respiratory abnormalities" heading

- 786.03—Apnea
- 786.04—Cheyne-Stokes respiration
- 786.05—Shortness of breath
- 786.06—Tachypnea
- 786.07—Wheezing

Injury and Poisoning

Code 965.6, "Poisoning by antirheumatics (antiphlogistics)," is now invalid with the addition of two new five-digit codes:

- 965.61—Poisoning by propionic acid derivatives
- 965.69—Poisoning by other antirheumatics

A new five-digit code (996.56—Mechanical complications due to peritoneal dialysis catheter) was added under the "Child maltreatment syndrome" heading.

A new five-digit code (996.68—Infection and inflammatory reaction due to peritoneal dialysis catheter) was added under the "Infection and inflammatory reaction due to internal prosthetic device, implant, and graft" heading.
Factors Influencing Health Status and Contact with Health Services

Code V02.5, "Carrier or suspected carrier of other specified bacterial diseases," is now invalid with the addition of three new five-digit codes:

- V02.51—Carrier or suspected carrier of Group B streptococcus
- V02.52—Carrier or suspected carrier of other streptococcus
- V02.59—Carrier or suspected carrier of other specified bacterial diseases

A new code (V10.48—Personal history of malignant neoplasm of epididymis) was added under the "Personal history of malignant neoplasm of genital organs" heading.

Code V13.6, "Personal history of congenital malformations" is now invalid with the addition of two new five-digit codes:

- V13.61—Personal history of hypospadias
- V13.69—Personal history of congenital malformation

Code V16.5, "Family history of malignant neoplasm of urinary organs," is invalid with the addition of two new five-digit codes:

- V16.51—Family history of malignant neoplasm of kidney
- V16.59—Family history of malignant neoplasm of other urinary organs

Code V18.6, "Family history of kidney diseases," is now invalid with the addition of two new five-digit codes:

- V18.61—Family history of polycystic kidney
- V18.69—Family history of other kidney diseases

A new four-digit code (V56.2—Fitting and adjustment of peritoneal dialysis catheter) was added under the "Encounter for dialysis and dialysis catheter care" heading.

A new five-digit code (V58.62—Encounter of aftercare for long-term (current) use of antibiotics) was added under the "Long-term (current) drug use" heading.

Source: American College of Physicians – American Society of Internal Medicine

(URL: http://www.acponline.org/journals/news/dec98/icd9cm.htm)


BIOGRAPHICAL SKETCH

Balaji Krishnaprasad was born in Chennai (Madras), India, on the 21st of September 1978. He completed his bachelor’s in computer science and engineering from the University of Madras in May of 2000. He joined the University of Florida to do his master’s in computer and information sciences in the Fall 2000 semester and has been working under the supervision of Dr. Douglas Dankel on the ICDNeT project.