5-2013

A Medical Data Cleaner

Jahnavi Yetukuri

A MEDICAL DATA CLEANER

by

Jahnavi Yetukuri

A report submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Science

UTAH STATE UNIVERSITY
Logan, Utah

2013

ABSTRACT

A Medical Data Cleaner

by

Jahnavi Yetukuri

Utah State University, 2013

Major Professor: Dr. Stephen Clyde
Department: Computer Science

This report describes medical-data cleaning tool, called *MedDataCleaner* that can detect outliers in medical data and assistant Database Administrators in resolving data-related problem. Specifically, *MedDataCleaner*, enables the users to define cleaning rules and offers the ability to choose classification methods that help determine if the data is good or bad. *MedDataClearer* uses Vitruvian DB objects for object-relation mapping (ORM) support and Vitruvian alignment links for designing the GUI.

My contribution towards this work includes designing the user interfaces using Vitruvian Alignment links, design and implement mean, standard deviation and neural classification methods using Vitruvian DB objects.

(50 pages)

ACKNOWLEDGMENTS

I extend my deepest gratitude to my advisor Dr. Stephen Clyde for his support, guidance and valuable suggestions. Dr. Clyde is a person whom everyone would love to work with. I am glad that I learned the basics of Object Oriented Design under a person who masters in Object oriented design. Dr. Clyde is and remains my best role model for a teacher and mentor. I am grateful for his insightful discussions and encouragement, which helped me resolve several design issues. Without his guidance and persistent help this dissertation would not have been possible.

I am grateful to my committee members, Dr. Curtis Dyreson and Dr. Nicholas Flann for their interest in this project. I would like to extend my appreciation to Abhinav Nahar, a fellow colleague for his insightful thoughts on design and co-operation, especially Brian Smith (Vitruvian framework developer) for his contribution towards resolving Vitruvian issues quickly and for a wonderful framework like Vitruvian which made most time consuming tasks like designing user interfaces easier.

I especially thank my family, Haranathbabu Yetukuri, Subbalakshmi Yetukuri, Karthik Yetukuri, Saikiran Panchakarla, Harisha Yetukuri and friends Udara Weerakoon, Prabhanjali and Shreeramkumar for their unconditional love, support and care. I would not have made it this far without them.

Jahnavi Yetukuri

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Quality data is an important asset to modern organizations, particularly as they as become more dependent on inter-organizational or multi-sourced data. Unfortunately, many organizations suffer from "dirty data," which includes incomplete records, missing field values or inter-record inconsistencies. Cutter Consortium, an IT advisory firm, identified in a report the following common sources of dirty data [5]:

1. poor data entry, which includes misspellings, typos and transpositions, and variations in spelling or naming, constitutes major sources for dirty data.
2. missing data from database fields.
3. lack of companywide standards in data coding.
4. mismatched syntax, formats and structure, e.g. variation in the number or type of name fields and different phone number formats.

According to the Data Warehousing Institute (DWI), the cost of bad or dirty data exceeds $600 billion annually [1].

In the medical field in particular, dirty data can cause increased costs, inefficiencies, liability risks and degraded quality of care [6]. Further, such dirty data causes significant and immediate need for a solution because it can lead to medical errors. The Institute of Medicine estimated in its report that around 44,000 to 98,000 lives are lost every year as a result of medical errors in hospitals [15]. Therefore, dirty data is motivating people all around the world to develop data-cleaning methods to handle bad data in medical health data. Data cleaning or data scrubbing is the act of detecting and

correcting (or removing) corrupt or inaccurate records from a record set, table, or database [7]. The data-cleaning process involves classifying data, detecting missed values, detecting and removing redundant records, checking if the data within the databases is accurate, and identifying the data that is outside the expected range.

In this project, I developed a medical data cleaning tool, called *MedDataCleaner*, using Vitruvian DB objects for object-relation mapping and Vitruvian alignment links for organizing and constraining the user-interface layout. Vitruvian DB objects and alignment links are discussed more in detail in Chapter 3 and Chapter 4.

This tool provides an interface for the Quality Assurance (QA) and Database (DB) users to connect any health database (local or remote) that they want to clean. After connecting to the database, the users can load the basic schema information pertaining to the database such as database name, table names, column names, and their data types. These schemas help users analyze the health-care data in a database and classifying them into categories, such as good, bad, expected, uncommon, normal, and optimal. Users can define cleaning rules for specific data types, based on the need, by doing the following three tasks:

1. Define a domain specifies the unit and the data type of the data to be cleaned
2. Select the data-value classification method that fits the domain and will allow the user to provide the most accurate means of characterizing the data as good, bad, expected, uncommon, normal, and optimal. These classification methods are as follows: 1) range, 2) format, 3) mean, 4) standard deviation, 5) discrete value and 6) neural classification method. See Chapter 2 for more details.

3. Determine unit conversions that map data in a single domain but with multiple units of measure to a data with a single unit. For example, if domain for birth-weight value may include data measured either in ounces or grams. The conversation rule could map all birth-weight data measured in ounces to birth weights measured in grams.

The data-cleaning services available in MedDataCleaner include:

1. Identification of missing data: Identifies the null values existing within the medical databases.
2. Identification of outliers
3. Identification of unusual (yet syntactically correct) values. (e.g. 999 or 9999)

Chapter 2 discusses user goals for a MedDataCleaner tool, summarizes a structural analysis, and lists the system's functional requirements. Chapter 3 explains the design of *MedDataCleaner* in terms of user interfaces, application-layer components, and database structures. Chapter 4 provides a brief background on the technologies used and some of the implementation challenges. Chapter 5 discusses software testing to verify the correctness of the implementation of *MedDataCleaner*. Finally, Chapter 6 gives some ideas about the possible extensions for the project and future enhancements.

CHAPTER 2

SYSTEM ANALYSIS

This chapter documents the functional requirements for a medical data cleaning tool using Unified Modeling Language (UML) diagrams[1]—e.g., UML use-case and class diagrams. Use-case diagrams visualize, specify, and document the behavior of a system. In a nutshell, the use-case diagrams in Section 2.1 provide developers with a high-level overview of what a MedDataCleaner should do. The class diagrams in Section 2.2 specify the structural makeup of the system [12]. Class diagrams describe the key objects and their relationships in a system. Class diagrams define three perspectives that help developers solidify the design of a system: conceptual, specification, and implementation. Section 2.3 includes the functional requirements. Functional requirements specify the functionality or specific behavior of the system.

2.1. User Goals

User goals are captured using use-case diagrams. Use-case diagrams consist of use cases and actors. An actor represents a set of roles that users play when interacting with the system. Actors can be human or can be automated systems [14].

Key actors identified in the MedDataCleaner include quality assurance users and database (DB) owners; their interactions are shown in Figure 1. Quality assurance users

[1] UML or the *Unified Modeling Language* helps specify, visualize, and document models of software systems, including their structure and design. It includes use-case diagrams, class diagrams, interaction diagrams, state charts, activity charts, and more. UML can also be used for business modeling and modeling of other non-software systems. Readers who are unfamiliar with UML can refer to any of the many textbooks on the subject, or the official specification published by the *Object Management Group* (OMG).

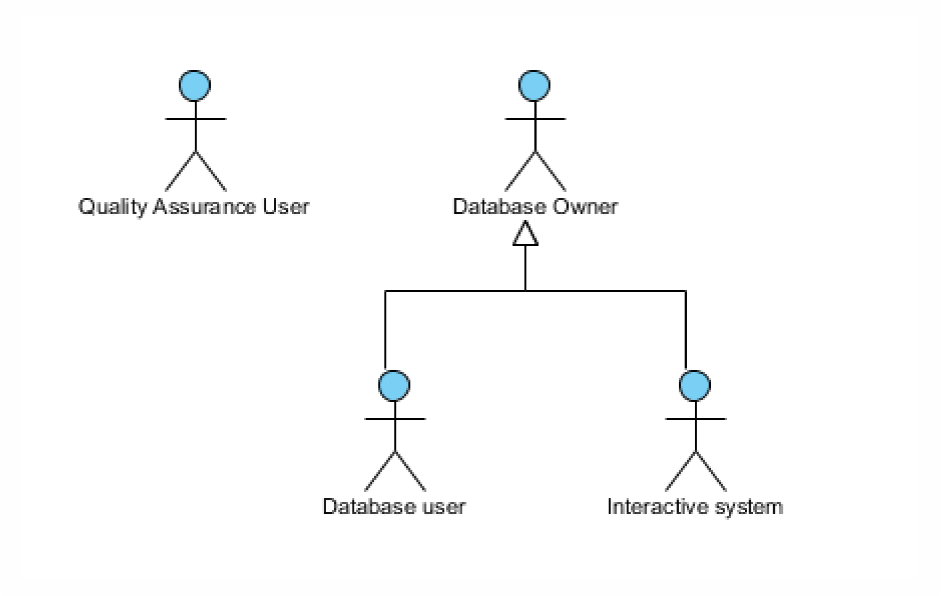

Figure 1: Actors in MedDataCleaner

play the role of defining cleaning rules for columns in the database tables. DB owners can be human or electronic systems. DB owners are further classified into DB users and interactive systems. DB user is responsible for maintaining a database and therefore performs various cleaning activities on the data. An interactive system also cleans the data in the databases it interacts with. The only difference between interactive systems and DB users is DB users are human and interactive systems are electronic systems.

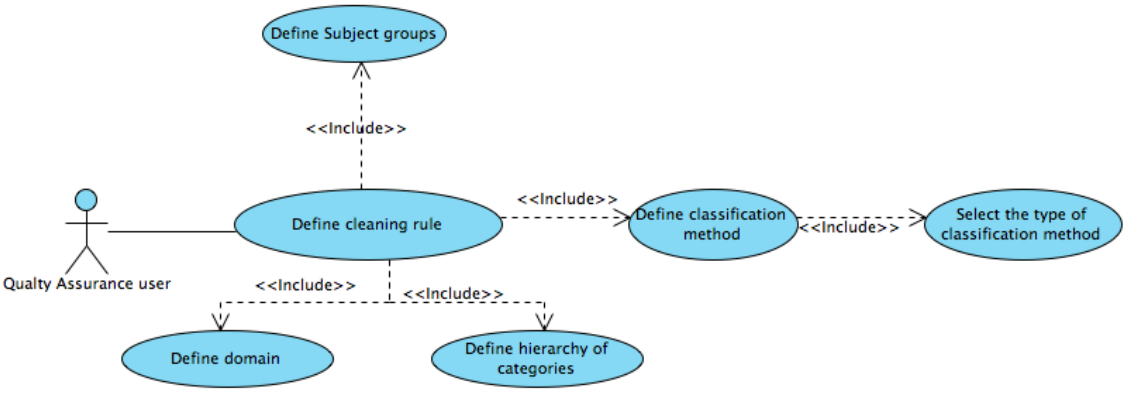

Figure 2: Use-case diagram describing the goals of Quality Assurance user

The use-case diagram in Figure 2 captures the goals of Quality Assurance users. QA users define cleaning rules and the cut-off values that the classification methods use to classify the data. To enable cleaning, each cleaning rule has to be provided with additional information such as subject group, domain, classification methods, and category hierarchy.

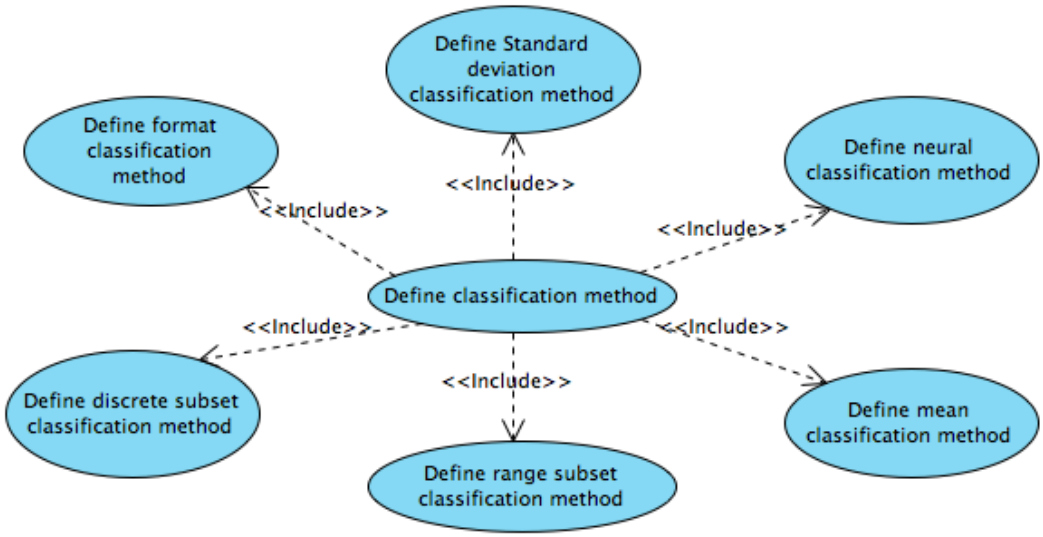

Figure 3: Usecase diagram depicting the types of classification methods a user can define

Figure 3 shows the sub goals of the general goals of defining classification methods. A user can choose to define any of the following classification methods: discrete value, range subset, standard deviation, mean, format, and neural classification methods. These classification methods classify the data into categories based on cutoffs specified by the user.

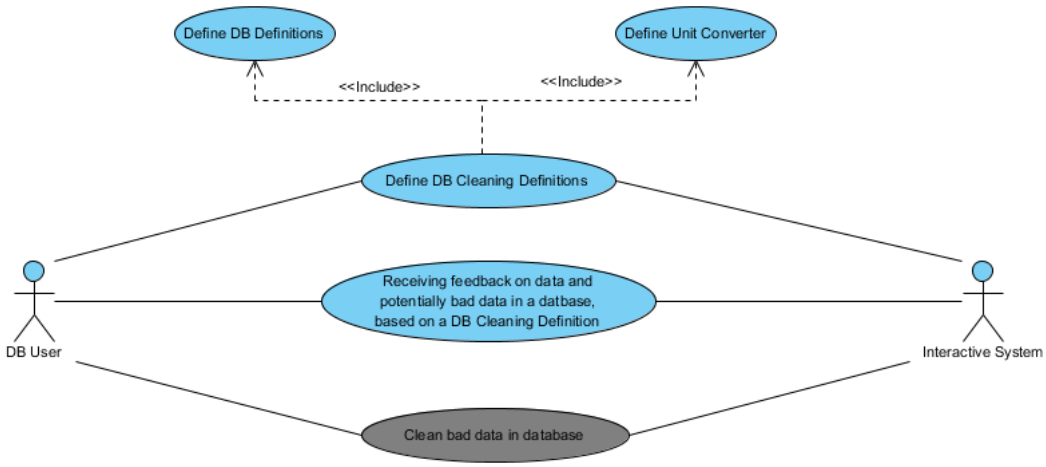

Figure 4: Usecase diagram describing the goals of DB user and interactive systems

In Figure 4, the use-case diagram captures the goals for a database user or an information system. To define cleaning definitions for the database, a user has to give a database definition and define a unit converter. Giving a database definition includes specifying the connection string. Unit converter is used when the unit defined for the cleaning rule is different from the unit defined for a column. After performing classification of data, the MedDataCleaner provides statistics to the user. These statistics enables the user to analyze data. The use case "Clean bad data in a database" is filled in gray because this project emphasizes data classification with data cleaning as future work.

2.2 Structural Analysis

Figures 5 and 6 are UML class diagrams that describe key object classes for detecting the data problems and helping user correct those problems.

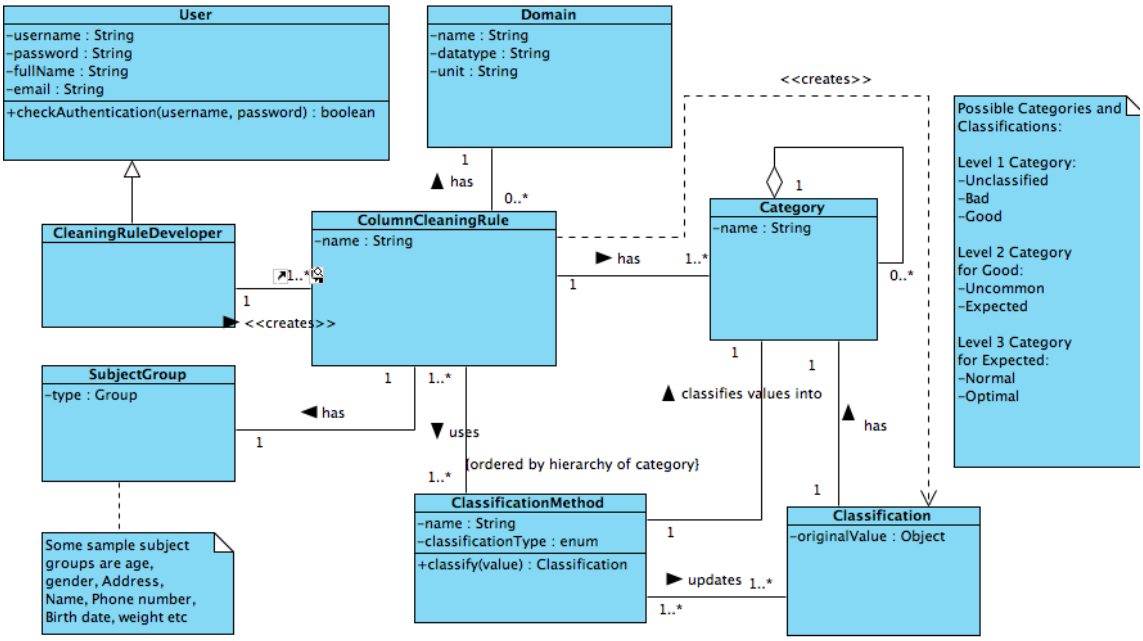

Figure 5: Class diagram showing cleaning rules and other important classes

From a preliminary analysis of sample sets of medical data, I was able to observe the following general characteristics:

- Medical data values collected from laboratories, hospitals, and personal health systems supposed to fall with some range of expected values or come from a set of possible discrete values.
- The data ranges may vary by gender. For example, the threshold for high-density lipoprotein (**HDL**) cholesterol in men is < 40mg/dL = low and in women is < 50mg/dL = low.
- Data sets often include outliers. Analyzing the data using the mean and standard deviation enables users to track values that are erroneous and also provides users with statistics on the overall data.

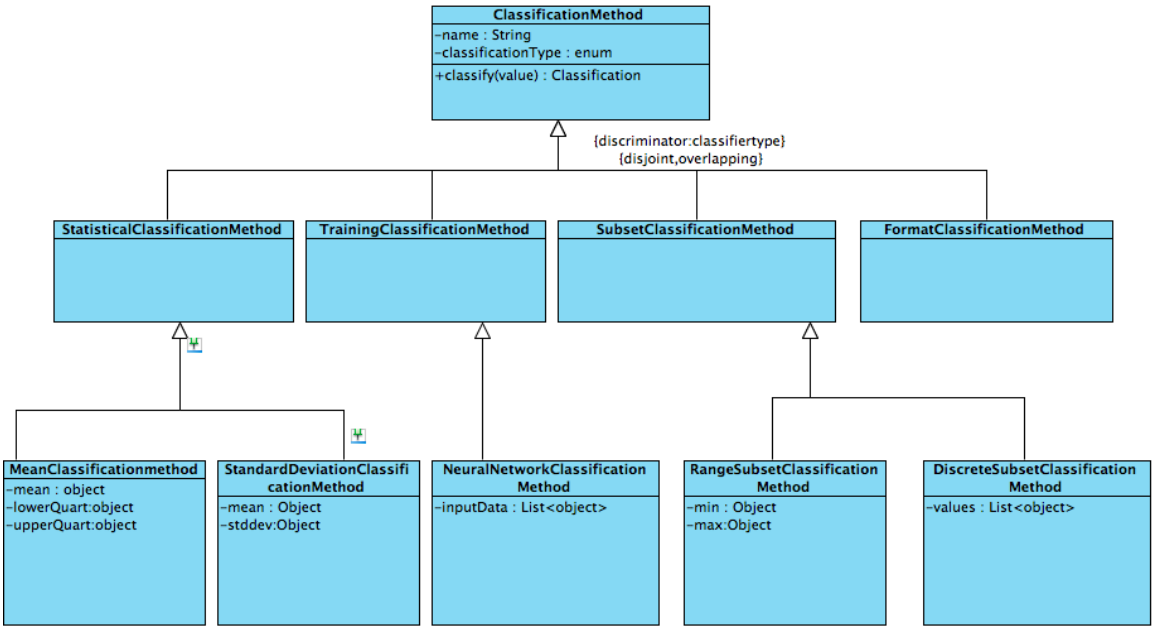

Figure 6: Class diagram depicting the classification method classes

I addressed the above observations by modeling different kinds of data-value classification methods, see Figure 6.

2.2.1 Classification Methods:

The base class *Classification Method*, represents the set of all possible data-value evaluation schemes. This class represents objects that classify data values into categories. Medical data is not always numerical data; it also includes personal information, such as date of birth and phone numbers, so a single classification method is not sufficient. Therefore, the *Classification Method* class is then partitioned into four specializations:

i) *Statistical Classification Method:*

Statistical methods help identify outliers in a given set of data. *Statistical Classification Method* is further divided into *Mean Classification Method* and *StandardDeviation Classification Method. Mean Classification Method* uses

lower, middle and upper quartile to classify. StandardDeviation Classification Method uses combination of mean and standard deviation to classify data.

ii) *Training Classification Method:*

This method uses nprtool, neural network pattern recognition tool to detect the outliers

iii) *Subset Classification Method:*

This Classification method is further divided into *Range Subset Classification Method* and *Discrete Subset Classification Method.* Range subset Classification Method classifies data into specified range. Discrete Subset Classification Method handles medical data with special values like 999, 777

iv) *Format Classification Method:*

This is a classification method based solely on data-value syntax. For example, SSN, DOB etc

The classification methods classify the data into categories, namely special meaning, good, and bad. Good values are subdivided into common and uncommon. Common values are again classified into normal and optimal.

2.2.2 Column Cleaning Rules:

The column-cleaning rules consist of a hierarchy of categories, See Figure 5. Defining a cleaning rule for a column involves selecting a classification method and assigning the classification method to the category. Now, the classification method classifies the data in to specified category. This structure enables the classification method to interact with other classification methods. For Example, for defining cleaning

rule for Cholesterol column, StandardDeviation Classification Method is used. This method classifies data into one of the categories by calculating mean - 3*standard deviation and mean + 3*standard deviation. The output data gathered using this method could be fed to other classification method to get a finer classification. For, example, output from this can be fed to Range classification method, which further classifies the data into other category.

Domain contains the unit and data type. Each cleaning rule has a domain associated with it. For defining the cleaning rule for a column, the domain defined in the cleaning rule has to be compatible with unit and data type of the column.

2.2.3 Unit Converter:

Unit conversion is important as medical data is represented in different units across the world. Also, The unit defined for the cleaning rule may be different from the column unit (e.g., unit of lab values), but these units can still be compatible. For example, the column to be cleaned is considered to be serum cholesterol and the unit for the column is reported as milligrams per deciliter (mg/dL). But the cleaning rule uses the unit millimole per liter (mmol/L). Though these units seem different, they are compatible as mmol/L is the SI unit for serum cholesterol while mg/dL is the conventional unit. So, instead of creating a new cleaning rule, the existing cleaning rule can be used by converting the unit using Unit converter class. See Fig. 7.

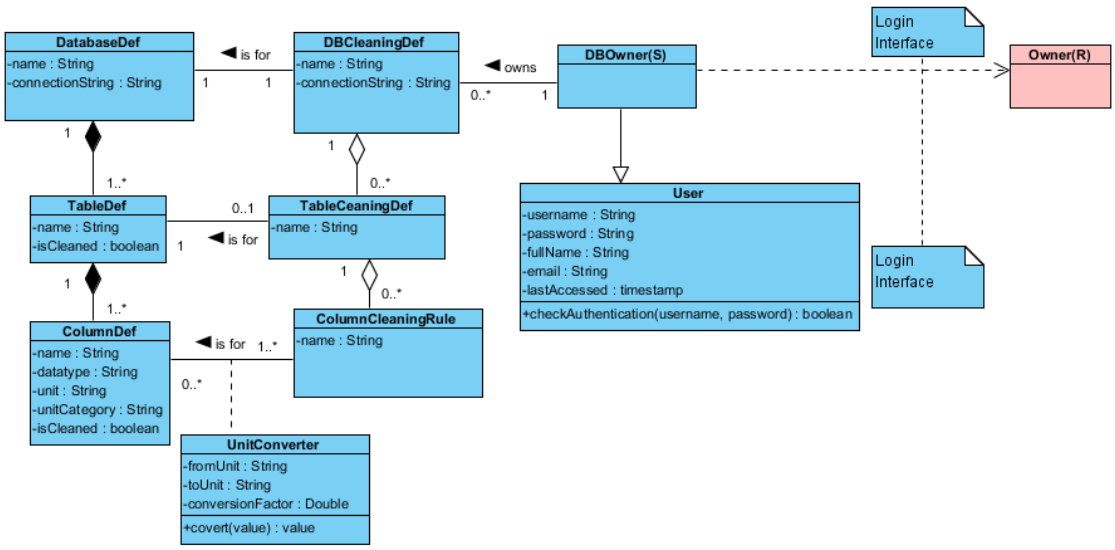

Figure 7: Class diagram showing database-related classes

2.3 Functional Requirements

The functional requirements capture the core functionality of the application. This section includes functional requirements for MedDataCleaner:

2.3.1 Usability

MedDataCleaner must provide user-friendly interfaces to the user, which enables the user to add the database, analyze the data within database, and display the statistics for the analyzed data.

2.3.2 Accessibility

Only users with a legitimate username and password will be able to access the MedDataCleaner.

2.3.3 Navigation

Users should be able to navigate easily among the cleaning rule, domain, and classification method forms.

2.3.4 Functionality

This section describes the functionality in detail for *MedDataCleaner:*

2.3.4.1 Add a database

Users should be able to add a new local or remote database by specifying the data source name (DSN) and the connection string.

2.3.4.2 Testing the connection and loading the database

Users should be able to test if the database connection is established successfully or not. If successfully established, it loads the tables and columns of the database.

2.3.4.3 Defining cleaning rule

This tool should enable users to define, save, edit, and delete a cleaning rule. The program should store the following information for a cleaning rule:

- Name: Name of the cleaning rule cannot be null or empty

- Subject Group: Subject group is optional. Some sample subject groups are age, gender, etc.

2.3.4.4 Defining classification method

This tool shall enable the users define a classification method. Classification methods should classify the data values into categories. A classification method consists of the following information:

- Name: Classification method name cannot be null or empty

- Type of Classification method: This tool allows the user select a classification method among range subset, discrete, standard deviation, mean, format, and neural classification methods.

2.3.4.4.1 Mean deviation classification method

When used for classification, this classification method calculates the mean for the data and enables the user to classify the data based on mean.

2.3.4.4.2 Standard deviation classification method

When used for classification, this classification method classifies the input data based on both mean and standard deviation (i.e., the standard deviation factor).

.2.3.4.4.3 Neural classification method

This classification method classifies a column value into a category using the nprtool (neural network pattern recognition tool).

2.3.4.4.4 Range subset classification method

When used for classification, the range subset classification method should store the maximum and minimum values. At least one of the minimum or maximum values should not be null or empty.

2.3.4.4.5 Discrete subset classification method

When used for classification, the discrete subset classification method should store all discrete unique values.

2.3.4.4.6 Format classification method

When used for classification, the format classification method should store the format. Format cannot be null or empty.

2.3.5 Defining domain

This tool will enable users to save, edit, and delete. Users should be able to define a domain, which consists of the following information:

- Name: Domain name cannot be empty or null.
- Data-type: Data type cannot be null and should be among the following: numeric, string, Boolean, or timestamp.

- Unit category: Unit category is optional. Unit category represents the measuring criteria. For example, length, volume, speeds, etc.

- Unit: For a given unit category, there exists a set of units. For example, unit category length has meter, centimeter, and millimeter as units. Unit is optional.

2.3.6 Performing unit conversion

Unit converter should enable the user to convert the column unit to the cleaning rule unit. It stores the following information:

- From unit: From unit cannot be null or empty.

- To unit: To unit cannot be null or empty.

- Conversion factor: cannot be null or empty.

2.3.7 Generate statistics of data

This tool shall enable the user to generate overall statistics of the data analyzed i.e. count of the bad, good, expected, normal, optimal values within the input data.

CHAPTER 3

ARCHITECTURAL DESIGN

This chapter explains the architectural design for a software tool, called MedDataCleaner, that satisfies the requirements summarized in Chapter 2. I use UML package and class diagrams to communicate the architectural design because they focus on module organization and data structures [21].The architecture of the *MedDataCleaner* can be organized in three layers: user interface layer, application layer and database layer. Fig. 8 depicts the package diagram that captures all layers of the MedDataCleaner. The user interface package contains graphical user interface classes and windows forms that allow users to view program features. The application package includes classes that contain the functional logic for adding a database: defining, viewing, and editing cleaning rules, classification methods, and domains. Database layer represented as metadata layer in Figure 8 contains the DB-objects and DB-lists for all the tables in the database. The DB-objects and DB-lists are generated by Vitruvian DB-objects.

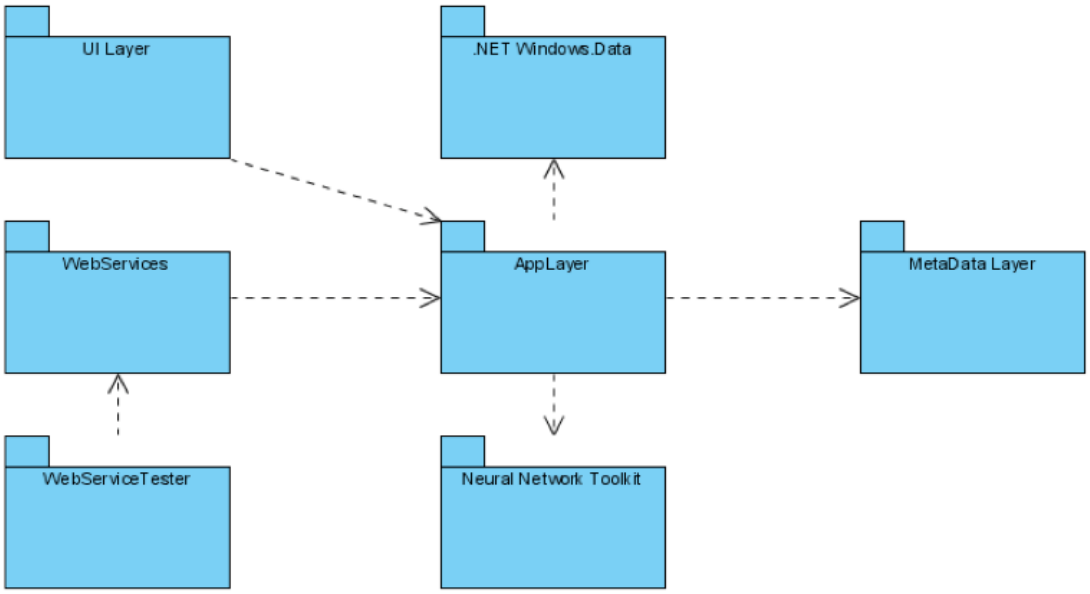

Figure 8: Architectural diagram of the MedDataCleaner

3.1 Front End Layer

3.1.1 User interface

Most of the software developed today is interactive software. Having a good interface makes things simpler for the user who is interacting with the software. Developing good interfaces increases loyalty and reduces support costs. The user interface for an application can make it or break it [16]. The usability of software depends on the user interface of the software.

Designing and implementing user interfaces for software requires a lot of time and effort. Moreover, the code written for implementing a user interface often makes up over half the total application code. The process of designing interfaces could be made easier and simpler for a programmer if there are some additional features in the development environment which could help the programmer maintain right–left alignment, visual closure, etc.

The .Net GUI development environment supports placing and positioning of controls with respect to edges on forms using anchors, but it does not support positioning or resizing of controls with respect to each other. Therefore, if the position of a control in a form is changed, all the controls on the form have to be rearranged accordingly. Also, an extra amount of code must be written to attain liquid layout. All this accounts for additional development time and effort, which could be reduced significantly by linking controls graphically.

To overcome the above stated hurdles, I used the Vitruvian framework to design the GUI in this project. Section 3.2 discusses this more in detail.

3.1.2 Background on the Vitruvian Framework

The Vitruvian framework supports several techniques for assisting the programmers with graphic user interface (GUI) development, including alignment links, which enable developers to constrain the size and position of GUI components; layouts, which allow them to construct GUIs programmatically; and templates, which capture common layout patterns for users.

The *alignment links* feature enables developers to constrain the size and position of GUI components directly in Microsoft's Visual Studio Form Designer without needing to write any code by hand. Alignment links provide the following six ways to link controls together:

1. *Position* On positioning the control, maintains the relative distance with the parent.

2. *Resize* On resize, maintains the relative distance with the parent.

3. *Percent* By moving the control, maintains the relative distance with the parent, as a percentage.

4. *Center* Moves the control so that it is centered with the parent.

5. *Same Size* Resizes the control so that it is the same size as the parent.

6. *Percent Size* Resizes the control so that it maintains the relative size, as a percentage of the parent.

A developer must first add *AlignLinkDesigner* to Visual Studio to enable the alignment link feature. Now, when a control is selected, the *AlignLinkDesigner* displays a number of glyphs on the edges and vertices of the control. See Figure 9. To create an alignment link, select the child control, click one of the align-link glyphs and then finally click on the target areas that indicate probable link points. To remove an alignment link, click on the align-link glyph.

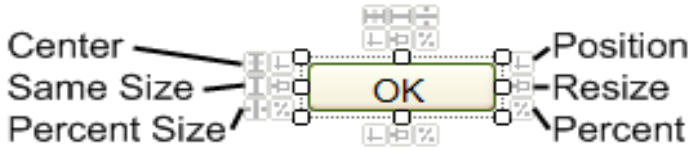

Figure 9: Six different align-link glyphs

Some common uses and examples of alignment links are shown below:

In example 1, the set of alignment links shown in Figure 10 causes the label to maintain the spatial relationship with the text box. This relationship is maintained even if the label text changes, the label font changes, or if the text box is moved or resized.

Figure 10: Position align-link glyph for a text box and label control.

In example 2, the alignment links shown in Figure 11 make the *Delete* button maintain the same size as the *Add* button, and they also maintain the spacing between the two buttons.

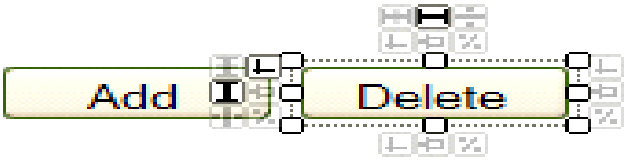

Figure 11: Example of maintaining same size on two button controls

In example 3, the resizing alignment links in Figure 12 attach the box to the edge of the form. When the form is resized the box will also be resized to maintain the distance relationship with the edge of the form.

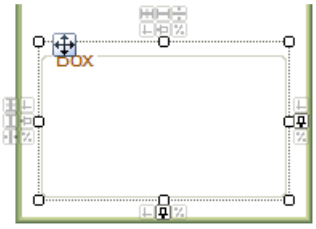

Figure 12: Resize box with the form

The alignment link designer adds the following graphics in Fig. 13 to the bottom edge of the control. The eye graphic toggles the visibility of the alignment links for the selected control. The lock graphic can be used to deactivate and activate alignment links for the

selected control. The trash graphic can be used to remove all alignment links from the selected control.

Figure 13: Eye, lock, and trash graphics

3.1.3 MedDataCleaner's User interface Design

The user interface package contains classes and windows forms. These forms enable the user to perform the following tasks:

- Log in securely
- Add new database
- Select a database for data cleaning
- Assign unit for a column
- Perform unit conversion
- Define/edit domain
- Define/edit classification method
- Define/edit cleaning rule

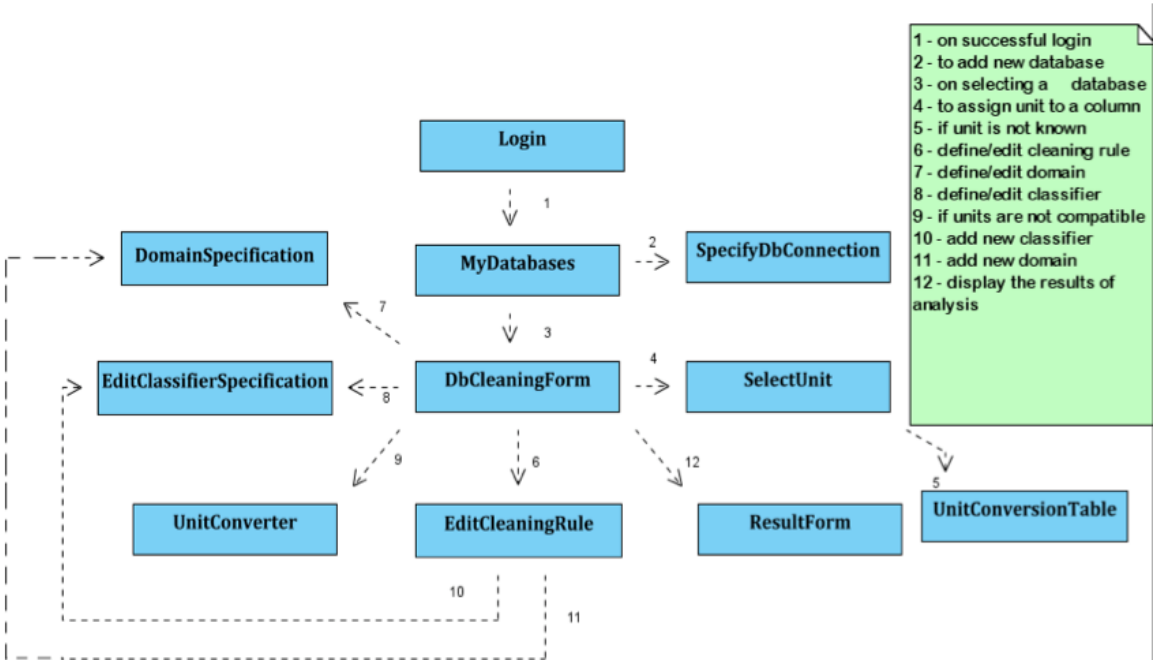

Figure 14: User interface navigation

Figure 14 shows the forms that comprise *MedDataCleaner's* user interface the possible user navigations betweent he forms. Below is a description of windows forms that I designed using alignment links.

The *Login* form enables users with legitimate usernames and passwords to log in and gain access to services offered by MedDataCleaner. In this form, I used the alignment link described in Examples 1, 2, and 3 to position the text boxes and labels (e.g., username and password), to make the login button have same size as cancel button, and to resize the picture box with form. Also, I resized the text boxes corresponding to username and password with the form. Similarly, I used the alignment links for position, resize, center, and same size for all the forms described below.

The *MyDatabases* form displays a list of databases added along with some additional details such as when it was accessed last and connection string. Also, it enables users to navigate to SpecifyDbConnection. The *SpecifyDbConnection* form enables users to add a new database. Users can add a new database by specifying data source name, driver, and the connection-string parameters. Users can even test if the connection is established successfully or not.

The *DbCleaningForm* is the most important form in the entire application and plays a significant role. Users can perform multiple tasks from this form, such as defining and editing cleaning rules, domain, and classification method.

The *SelectUnit form* enables the user to assign the unit for a column. If the user is not aware of unit then the user can navigate to the *UnitConversionTable* form by clicking on the link provided.

The *UnitConversionTable* form shows user a list of medical components along with their SI, conventional units, and unit conversion factor. This form even enables the user to search for the unit of the medical component by entering the name of the medical component.

The *DomainSpecification* form enables the user to define and edit the domain.

The *EditClassificationMethod* specification form enables the user to define and edit the classification method.

The *EditCleaningRuleSpecification* form enables the user to define and edit the cleaning rule. Also, enables the user to define a new classification method and domain.

The *UnitConverter* does as its name implies. If the unit categories of the column and cleaning rule are the same but the units are not compatible, then this form enables the

user to perform unit conversion by specifying the unit conversion factor and the unit conversion rule.

The *ResultForm* form presents final statistics of the data to the user after analysis and classification.

3.1.4 My experience with alignment links

Alignment links are easy to learn and use. These made designing user interfaces simpler for me. In this project, alignment links provided remarkable benefits in the following areas:

a) Maintainability

Unlike in Visual Designer, I could easily make changes to the screens and controls after the initial development of the screens.

b) Usability

Alignment links is as usable as Microsoft Studio's Visual Designer and is equally easy to use and learn. Initially, while using alignment links I would get confused among the glyphs. But in no time I became very comfortable using the glyphs.

a. Productivity

When a UI control is moved I didn't have to make any explicit adjustments to other controls on the form. The controls would rearrange automatically as they are linked using alignment links.

b. Lines of code

Using alignment links, I have created forms that support liquid layout at runtime, thus reducing the lines of code.

3.2 Application layer

Robustness, flexibility, reusability, scalability, and maintainability of software mainly depend on the application service layer design.

This layer consists of implementation classes that describe all code behind the forms. These classes map the logic in the implementation classes, captured in the analysis to the view. The class diagram in Fig. 10 shows DatabaseDef and related classes.

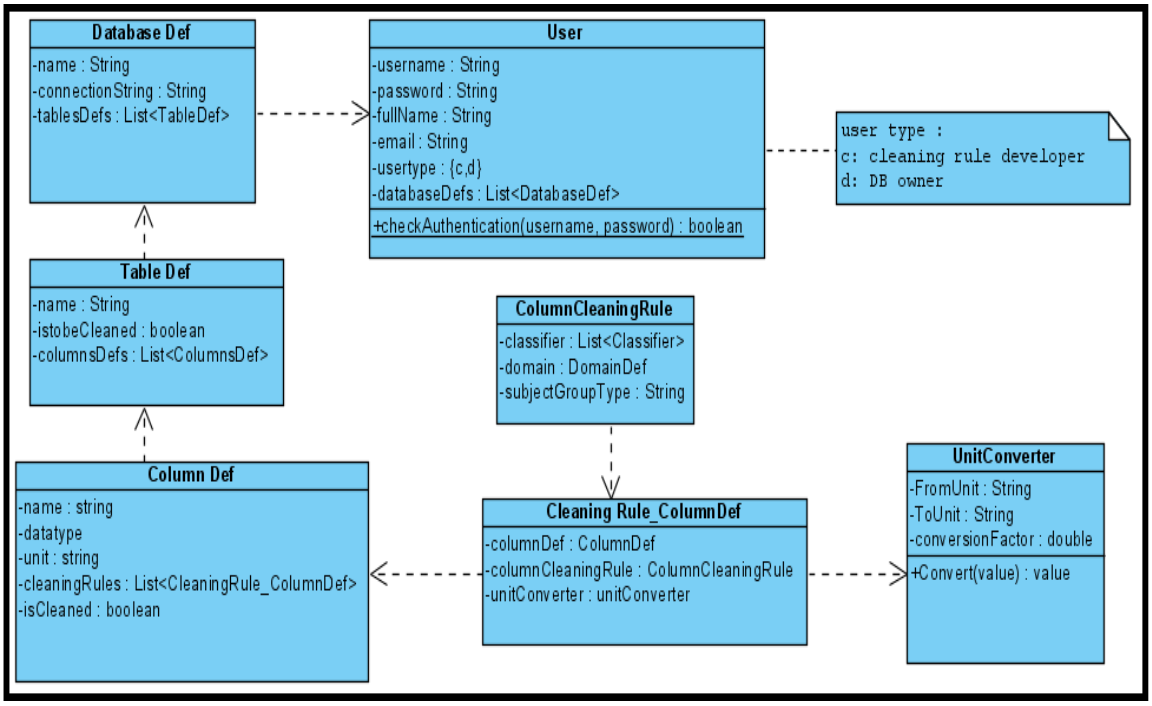

Figure 15: Class diagram describing Database Def

The user (QA user or DB owner) should first choose a database to clean. The tool enables the user to connect to the database using data source name or connection string

parameters and store the names of databases, tables, columns, data types, units, and connection strings. Since each user can have access to multiple databases, a user consists of a list of database def. Each database consists of multiple tables, which in turn consist of multiple columns. So, a *Database Def* object contains a list of *Table Def* objects and a *Table Def* object consists of a list of *Column Def* objects. *Column Def* object doesn't contain a list of values within the column; rather, it simply defines that column. Data values for a column are fetched as needed during the cleaning process. Medical data values have different data types, such as integer, float, timestamp, Boolean, string, etc. Also, medical laboratories and practitioners in the US use conventional units for reporting the results of clinical laboratory measurements, while those in other countries use the International Standard (SI) units.

To address this problem of different data types and units, domain is designed. Domain is defined for a cleaning rule and specifies the data type and unit of a column. A cleaning rule can be associated with a column only if the data type and unit of cleaning rule is compatible with that of the column. The MedDataCleaner uses numeric, string, date and time, and Boolean data types. If the unit categories of the column and cleaning rule are the same but units are incompatible, then unit conversion can be performed using the unit converter instead of creating redundant cleaning rules.

Unit conversion is another key component of the tool. Data reported in different units should fall into different numerical ranges. Therefore, if the ranges defined for SI units are used for data in conventional units, and then the resultant analysis becomes

faulty. In other words, unit conversion prevents wrong analysis of data. The unit converter is discussed in more detail in next chapter.

Cleaning Rule in Figure 11 is the most important class in the system. Most of the tool's functionality is embedded within cleaning rule. Each column to be cleaned has to have a cleaning rule associated with it. It is the cleaning rule that allows MedDataCleaner to classifying actual data values into categories. Each cleaning rule has a domain and classification-category hierarchy associated with it; the category hierarchy has a category which in turn has a classification method associated with it. The classification method is responsible for classifying the data within a column to the category it is associated with.

In a category hierarchy, it is not always necessary to have root categories. To address the cases without a root category, a subtree without a root structure is designed by my teammate. This structure enables the classification methods to be accessed hierarchically, i.e., the result of the parent category is passed as an input to a child category.

All classification methods (standard deviation classification method, mean classification method, format classification method, range-subset classification method) classify the data values into categories and update the classification. This is done using classify () and update () methods.

An example of a category hierarchy: Special values - If a value is classified under special values it cannot be considered as good or bad data. An example of special values is 999, 9999, which are often seen in medical databases. Good values- the values classified as good can further be classified as common and uncommon. Similarly, the values classified as common can further be classified as normal and optimal. A category

at each level is associated with a classification method. For example, a user can use the statistical classification method to classify the values into the "good" category and use the range subset classification method to classify the values in "good" into "normal." This hierarchy enables the data to be classified more efficiently.

The MedDataCleaner consists of the following classification methods:

1. **Standard deviation classification method:** This classification method classifies the values into categories based on the mean and standard deviation. The implementation of this classification method is discussed in detail in the next chapter.
2. **Mean classification method:** This classification method classifies a column value into a category based on its deviation from the mean, which is discussed in more detail in the next chapter.
3. **Neural classification method:** This classification method classifies a column value into a category using the nprtool (neural network pattern recognition tool).
4. **Range subset classification method:** Medical values can be classified as optimal or normal based on the range. So, this classification method checks if the column value is within the maximum and minimum range.
5. **Discrete subset classification method:** Medical values can be represented as single value instead of a range. So, this classification method checks if the column value is one of the discrete values.

6. **Format classification method:** This classification method checks if a column value is in a specified format or not.

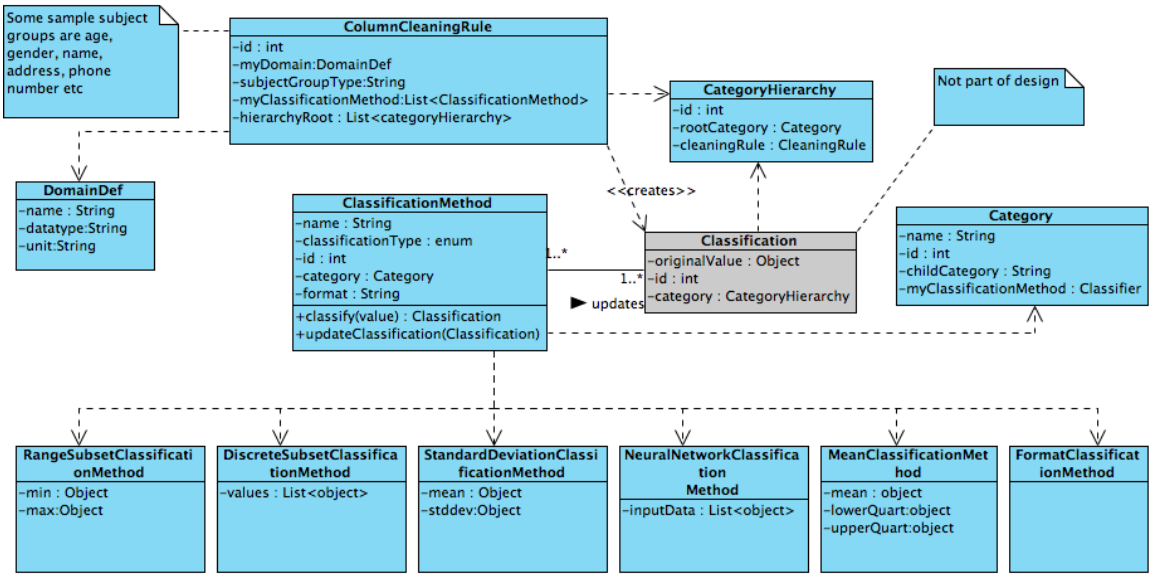

Figure 16: Class diagram showing database design of cleaning rule

3.3 Neural-network toolkit background and design

Neural Network Toolbox provides tools such as the pattern-recognition tool (nprtool), fitting tool (nftool), clustering tool (nctool) and time-series tool (ntstool) for designing, implementing, visualizing, and simulating neural networks. Neural networks are used for applications where data sets to be examined are large and formal analysis would be difficult, such as pattern recognition [18]. The toolbox supports implementations for feed-forward networks, radial-basis networks, dynamic networks, and self-organizing maps.

The MedDataCleaner uses the neural network pattern recognition tool for classifying data into categories. This is discussed more in detail in the next chapter.

To connect to and access neural network toolkit via C# the following steps have to be followed:

1. In C#, navigate to Project ->Add Reference, then select the COM tab.
2. Under COM tab, select Matlab application.
3. Use private MLApp.MLAppClass matlab to create a C# Matlab object.
4. Now, code can be executed in matlab via C# using matlab.Execute("Matlab code")

CHAPTER 4

UNDERLYING TECHNOLOGY AND IMPLEMENTATION DETAILS

To implement *MedDataCleaner*, I used C#.NET 2005, PostgreSQL 8.3, Vitruvian DB objects for ORM support, and Vitruvian alignment links for designing the GUI. Since Vitruvian is not fully compatible with C#.NET 2008, we used C#.NET 2005. Assessing the usability and improvement of Vitruvian-DbObject and alignment links is a secondary goal of the project. Alignment links are discussed in Chapter 3.

4.1 Introduction to Vitruvian DB-Objects

Object-relation mapping (ORM) maps objects in an object-oriented system to the data stored in a relational database. DB-Objects are similar to ORM.

DB-Objects are used to represent a relational model in an object model. A table in a relational model maps to a class in an object model, and a column in a relational model maps to properties of class in an object model.

Data within the databases passes via some data entity before being displayed on a user interface. See Fig. 17. After editing, the data is transferred to the data entity where it is stored in the database.

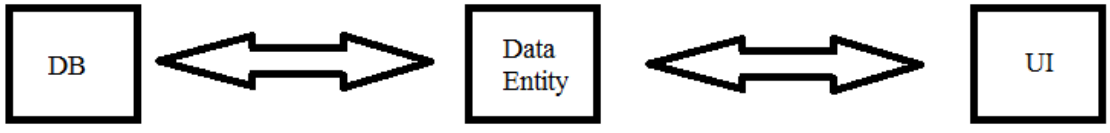

Figure 17 : Relationship between DB, data entity and UI

DB-Objects provide the following features:

1. Load(): Load the data into the DB-Object.
2. Reload(): Load the new set of data from database.
3. Save(): Save the DB-Object to the database.
4. Delete(): Delete the DB-Object from the database.
5. ResetValues(): Reset the properties of a DB-Object.
6. RelationalSave(): Save the DB-Object and the children tables of the current DB-Objects.
7. RelationalDelete(): Delete the DB-Object and the children tables of the current DB-Objects.

DB-Object also keeps track of its current state. A DB-Object can be in one of the following states:

1. New: DB-Object is just created and not saved into the database.
2. Synced: DB-Object is in synchronization with the database.
3. Modified: DB-Object one of whose properties has been modified and not saved in the database.
4. Deleted: DB-Object is deleted.
5. Detached: DB-Object exists in the database but is marked for deletion.

DB-Objects has a data wizard that automatically converts tables into classes and columns into properties. It is even capable of distinguishing between one-to-one and one-to-many relationships. The user can customize class names, properties, and relationships when generating DB-Objects.

4.2 Implementation details

This section discusses the implementation details of the MedDataCleaner. It also covers the challenges faced and solutions.

My contribution and the features I implemented in this project are:

1) **Graphical user interface:** I designed the user interfaces for the MedDataCleaner using alignment links, which is discussed in detail in Chapter 4.

2) **Connection string:** Adding a database can be done either by using DSN or by building a connection string. To build the connection string, connection-string parameters are needed. To build the connection string, I referred to the connection strings of various database servers including PostgreSQL, Oracle, SQL Server 2008, etc. at www.connectionstrings.com. Each of the database servers had several connection strings. For example, PostgreSQL contains the following connection strings:

i) Standard

Server=127.0.0.1;Port=5432;Database=myDataBase;User Id=myUsername;Password=myPassword;

ii) Command timeout setting

Server=127.0.0.1;Port=5432;Database=myDataBase;User Id=myUsername;Password=myPassword;CommandTimeout=20;

The CommandTimeout parameter is measured in seconds and controls for how long to wait for a command to finish before giving an error.

iii) Connection timeout setting

Server=127.0.0.1;Port=5432;Database=myDataBase;User Id=myUsername;Password=myPassword;Timeout=15;

To add a PostgreSQL database, the user can select the driver and select a connection type, which could be either of the above, and the connection string parameters corresponding to a connection type are displayed automatically.

3) Unit converter: After adding a database, the user can select a medical data column to clean. If the medical column doesn't contain a unit, the user can assign unit. To assign a unit for a column, the user has to select the unit category the unit belongs to. Then all the units corresponding to the unit category are displayed as a list. For example for unit category height, the possible units are inch, foot, and centimeter. The unit field can even be left empty. If the user is not aware of the unit for the column, then I provided a link to search for the unit of that column.

If the unit for the column is not empty, then a cleaning rule can be defined for the column only if the units are compatible. For example, if the user is cleaning the data column for creatine, the unit for the column is milligram per deciliter (conventional unit for creatine), and the unit defined in the cleaning rule is micromole per liter (SI unit for creatine). In such cases, unit conversion avoids creation of multiple cleaning rules for the same column having different units. The unit of the column can be converted to the cleaning-rule unit by multiplying the data values within the column by a conversion factor of 76.26. However, the unit conversion can only be performed if the unit category of both column and cleaning rule are the same.

4) **Visual representation view:** After adding a database, if the user wishes to view a visual representation of the data column to be cleaned, this feature presents a scatter plot of the data to the user.

5) **Classification methods:** The design and implementation of the mean, standard deviation, and neural classification methods:

Mean classification method: In the previous version, I classified the data within the column as good or bad based on the mean value. But choosing mean value to classify the data is not an efficient approach because if the data contains outliers then the mean of the data is affected. For example, consider the following set of data:

3 4 5 7 21 199 1000 9999

The mean for the data is 1404.75. So, if we classify the data above mean as bad and the data below mean as good then the analysis is faulty, with only a single data point above the mean. I went through many mathematical methods and came up with the following method. In this new mean method, first data is sorted and then the lower, middle, and upper quartiles are calculated for the given data using the following formulas:

Lower quartile = ¼ *(n+1) where n is the number of elements in the data.

Middle quartile = median and

Upper quartile = ¾ *(n+1)

Now, the data between the lower and upper quartiles is considered as this data is not affected by outliers. To classify the data into finer details, the same

procedure is repeated for data below the lower quartile and above the upper quartile.

Standard-deviation classification method

Similarly, even for the standard-deviation classification method, if we calculate the standard deviation for the entire data set, the analysis might be faulty as the data might include outliers. So, the mean for the data is calculated using the method described in 5.3.1. The standard deviation of the data is calculated for the data between the lower and upper quartiles. Now to classify the data, calculate mean - 3*standard deviation and mean + 3*standard deviation. Here, 3 is the multiplication factor for the standard deviation. To classify data further, repeat the same steps by varying the multiplication factor.

Neural classification method

Input data is arranged as columns in a matrix, then the target vectors arranged so that they represent the classes to which the input data are assigned. The target vectors are calculated using statistical methods. Now use the nprtool (neural network pattern recognition tool) in MATLAB via C# and adjust the validation, testing, number of hidden neurons accordingly and set the epochs to 250. The confusion matrix represents the data classified correctly and the data classified incorrectly. The main challenge I faced while implementing this classification method was while debugging: if an exception occured in MATLAB, the error reported in C# was “COM exception,” and no

other details were provided about the exception. So I had to debug the program, first placing breakpoints in both C# and MATLAB and then integrating them.

Dealing with null values and data such as 999 or 9999

In all the classification methods discussed above, mean plays an important role. But since the data contains null values, if we take the mean of the data considering null values to be zeros then this would result in loss of information. The other approach is if we replace the null values with the average of other values the resultant data might be faulty because outliers affect an average. So, I considered replacing the null values with the mean of the data that falls between the lower and upper quartiles, as outliers do not affect this data.

This project addresses data analysis with data cleaning as an extension. The above-discussed method can be used as one of the methods for replacing null values or values such as 999 or 9999.

CHAPTER 5

SOFTWARE TESTING

Software testing is a process of validating and verifying the quality of a product to provide stakeholders with information about the benefits and risks at implementation of the software product [19]. To test the quality and usability of the MedDataCleaner, we performed unit testing, integration testing, and user-acceptance testing. Sections 5.1, 5.2, and 5.3 contain details about unit, integration, and user-acceptance testing, respectively.

5.1 Unit testing

Unit testing takes the smallest piece of testable software in the application, isolates it, and determines if it behaves as expected. Each unit is tested separately before being integrated into modules. A large percentage of defects are identified during unit testing [20].

Test cases are the basic building blocks of unit testing. Test cases are written by developers to determine the results that an implemented method produces on a wide set of inputs provided by the user.

In the MedDataCleaner, unit testing was performed for select unit, DB cleaning, unit-conversion table, unit converter, creating and editing domain, cleaning rule, mean, neural and standard deviation classification methods.

5.2 Integration testing

Integration testing is a logical extension of unit testing. In integration testing, two individual units already tested are combined into a component and tested. The idea is to test combinations of pieces and eventually expand the process to test all the modules with

those of other groups. Eventually all the modules making up a process are tested together [20]. Integration testing is performed in three ways: the top-down, bottom-up, and umbrella approaches.

For the MedDataCleaner, we followed a bottom-up approach, i.e., the lowest-level units were tested and integrated first. The units were integrated and tested in the following order:

1) Domain and cleaning rule
2) Classification method and cleaning rule
3) Column definition and cleaning rule
4) Column definition and unit converter
5) Column definition, unit converter, and cleaning rule
6) Column definition, cleaning rule, and classification method
7) Column definition, unit converter, cleaning rule, and classification method
8) Testing integration of neural network toolkit with C#

Testing was performed on real medical data. All the bugs encountered were resolved during testing.

5.3 User-acceptance testing

User-acceptance testing is a phase of software development in which the software is tested in the "real world" by the intended audience [20] and is usually done before the delivery of the product. MedDataCleaner was fully tested against the requirements

defined in the analysis and design stages for real medical data, and the tool worked efficiently in classifying the data into good, bad, null, common, and optimal values.

CHAPTER 6

CONCLUSION AND FUTURE WORK

The MedDataCleaner works efficiently in classifying data into various categories using cleaning rules, domain, classification methods, and unit converter. Unlike other tools, the MedDataCleaner enables the user to choose the classification method to be used to classify the data. Cleaning rules can be applied in the cleaning process. Domain and unit converter helps in handling different units and data types. The classification method classifies the values into categories.

Data cleaning tool discussed in this project is currently a GUI based tool; as a next step towards its development, it could be implemented as a web application. Currently, The MedDataCleaner cleans a column using a cleaning rule and saves the cleaning rule corresponding to the column. If the user wishes to clean another column, the tool should suggest to the user an existing cleaning rule, which could be used to clean the data within the column.

REFERENCES

[1] http://www.acainternational.org/products-the-cost-of-dirty-data-13422.aspx

[2] http://www.pharmopro.com/Articles/2009/04/Clinical-Data-Cleaning-and-Validation-Steps/

[3] http://www.kosmix.com/topic/Electronic_health_record

[4] DATA CLEANING A prelude to knowledge discovery by Jonathan I. Maletic Kent State University, Andrian Marcus Wayne State University

[5] http://www.computerworld.com/s/article/78230/Data_Scrubbing

[6] http://download.101com.com/pub/tdwi/Files/DQReport.pdf

[7] http://en.wikipedia.org/wiki/Data_cleaning

[8] http://www.slidefinder.net/d/data_cleaning/6697032

[9] Tools for Data Translation and Integration by S. Abiteboul, S. Cluet, T. Milo, P. Mogilevsky, J, Siméon, S. Zohar

[10] http://msdn.microsoft.com/en-us/library/aa905986(SQL.80).aspx

[11] Background issues on data quality http://etd.library.pitt.edu/ETD/available/etd-03232007-105346/unrestricted/ahklym_etd2007.pdf

[12] http://www.builderau.com.au/strategy/designprinciples/soa/Creating-class-diagrams-With-UML/0, 339028846, 339170202, 00.htm

[13] http://www.fatfreekitchen.com/cholesterol/cholesterol_units.html

[14] Unified Modeling Language User Guide, by Grady Booch, James Rumbaugh, Ivar

Jacobson

[15] http://md-jd.info/abstract/Institute-of-Medicine-Report.html

[16] http://www.ambysoft.com/essays/userInterfaceDesign.html

[17] http://www.mathworks.com/products/neuralnet/description1.html

[18] http://en.wikipedia.org/wiki/Software_testing

[19] msdn.microsoft.com/en-us/library/aa292197 (VS.71).aspx

[20] http://searchcio-midmarket.techtarget.com/sDefinition/0,,sid183_gci836031,00.html

[21]http://meusite.mackenzie.com.br/rogerio/the-unified-modeling-language-user-guide.9780201571684.997.pdf