Event-Driven Dynamic Query Model for Sleep Study Outcomes Research

by

SULABH JAIN

Submitted in partial fulfillment of the requirements For the degree of Master of Science

Thesis Advisor: Dr. Guo-Qiang Zhang

Department of Electrical Engineering and Computer Science
CASE WESTERN RESERVE UNIVERSITY

January 2012
CASE WESTERN RESERVE UNIVERSITY
SCHOOL OF GRADUATE STUDIES

We hereby approve the thesis of

SULABH JAIN

candidate for the Master of Science degree *

Committee Chair: ________________________________
Dr. Guo-Qiang Zhang, Ph.D.
Professor,
Department of Electrical Engineering & Computer Science

Committee: ________________________________
Dr. Samden D Lhatoo, MD, FRCP
Professor,
Department of Neurology

Committee: ________________________________
Dr. Satya Sahoo, Ph.D.
Assistant Professor,
Division of Medical Informatics

November 02, 2011

*We also certify that written approval has been obtained for any proprietary material contained therein.
# Table of Contents

Table of Contents ........................................ iii  
List of Tables ............................................ vi   
List of Figures ........................................... vii  
Acknowledgement ........................................... viii  
Abstract .................................................. x  

1 BACKGROUND ........................................ 1  
1.1 About PhysioMIMI .................................... 1  
1.1.1 VISAGE .............................................. 3  
1.2 Utilizing Annotations as Query Filter ............... 3  
1.3 Motivation .............................................. 3  
1.4 Current Work .......................................... 4  
1.5 Organization of Thesis .............................. 6  

2 SLEEP MEDICINE ..................................... 7  
2.1 Introduction .......................................... 7  
2.2 Polysomnography ................................... 7  
2.2.1 Components of Polysomnogram .................. 8  
2.3 Different Types of Sleep Stages .................... 10  
2.4 Sleep Disorders ....................................... 11  

3 PSG QUERY SYSTEM .................................. 12  
3.1 Use Cases .......................................... 12  
3.2 Overview of the Software .......................... 13  
3.3 Integrated Query Formulator ....................... 14  
3.3.1 Query Formulator ................................ 14  
3.3.2 Logical AND/OR Operation on Selectors ........ 15  
3.3.3 Rearranging Selectors using Drag and Drop .... 16  
3.3.4 Grouping/Ungrouping of Selectors ............. 16  
3.4 Patient Cohort Browser ............................. 17  
3.5 Signal Visualization Tool ......................... 17  
3.6 PSG Query System - An Extension To PhysioMIMI .. 18
B Ruby on Rails
   B.0.1 Recommended Reading

C Example Code
   C.1 Event Selector Widget
   C.2 Snap View Generator
   C.3 Signal Visualization
      C.3.1 Ajax Request Initiated From Client
      C.3.2 Controller Action Code On Server Side
      C.3.3 Graph Generation Using Highchart Library
   Bibliography
List of Tables

8.1 Software Feature Comparison ........................................ 38
8.2 Software Feature Evaluation ........................................ 39
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>PhysioMIMI Architecture</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Software Components</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>A Typical Hypnogram From A Young, Healthy Adult [21]</td>
<td>11</td>
</tr>
<tr>
<td>3.1</td>
<td>PSG Query System as an Extension to PhysioMIMI</td>
<td>13</td>
</tr>
<tr>
<td>3.2</td>
<td>User Interface Displaying Toggle Button For Logical Operation</td>
<td>15</td>
</tr>
<tr>
<td>3.3</td>
<td>User Interface Displaying And Operator</td>
<td>15</td>
</tr>
<tr>
<td>3.4</td>
<td>Selector Panel Showing Drag And Drop Icon</td>
<td>16</td>
</tr>
<tr>
<td>3.5</td>
<td>Displaying A Group Containing 2 Selector Panels</td>
<td>16</td>
</tr>
<tr>
<td>4.1</td>
<td>Part Of Data Model Representing Tables Used In PSG Query System</td>
<td>21</td>
</tr>
<tr>
<td>4.2</td>
<td>Generation Of Selector Panels From Database Model</td>
<td>22</td>
</tr>
<tr>
<td>4.3</td>
<td>Query Evaluation</td>
<td>22</td>
</tr>
<tr>
<td>4.4</td>
<td>Events And Associated Recording Channels</td>
<td>24</td>
</tr>
<tr>
<td>5.1</td>
<td>Generation Of Query Panel Using Selector Widget</td>
<td>27</td>
</tr>
<tr>
<td>5.2</td>
<td>(A) Selector Widget With All Elements. (B) Selector Widget</td>
<td>28</td>
</tr>
<tr>
<td>5.3</td>
<td>Integrated Query Formulator Workflow</td>
<td>30</td>
</tr>
<tr>
<td>8.1</td>
<td>Query Performance Comparison Chart</td>
<td>40</td>
</tr>
<tr>
<td>8.2</td>
<td>Signal Visualization Tool Performance</td>
<td>42</td>
</tr>
<tr>
<td>8.3</td>
<td>Server Response Time For Various Tasks</td>
<td>44</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

First and foremost I offer my sincerest gratitude to my advisor, Dr. G. Q. Zhang, who has supported me throughout my thesis with his patience and knowledge while allowing me the room to work in my own way. I attribute the level of my Master’s degree to his encouragement and effort and without him this thesis, too, would not have been completed or written. One simply could not wish for a better or friendlier advisor.

I would like to thank Dr. Samden Lhatoo for all his teachings in the field of neuroscience and his encouragement for technological innovations in the field of medicine. I would like to thank him for all his feedback in making the research achieve its goals.

I would also like to thank another committee member, Dr. Satya Sahoo, for providing guidance during various different stages of the research and helping me to better understand the requirements of the project and providing valuable feedback on multiple software features of the project.

In my daily work I have been blessed with a friendly and cheerful group of fellow researchers. I would like to thank my team members Van Anh Tran, and Zendrix Ng for working together on different phases of the research. My special thanks again to Van Anh Tran and Remo Mueller (PhD) for being great mentors and working through the details in resolving some important research issues.

My sincere thanks to our Project Manager, Sridevi Cherukuri, for providing guidance from time to time and helping resolve technical issues in quickest possible time. She has always been so friendly and appreciating.

I would further like to acknowledge the contribution of Department of Epilepsy (Case Medical School). It has immense contribution in setting the stage for me for technological research in the field of neuroscience. My true and sincerely thanks to Dr. Hans Lüders for giving me an opportunity of
the lifetime of being associated with the department and believing in my potential. And a very special thanks to John Turnbull (PhD) and Farhad Kaffashi (PhD) for being great mentors and making me understand the technological challenges in the field of neuroscience.

Finally, I thank my parents for supporting me throughout my studies at Case Western Reserve University. Thank you all.
Event-Driven Dynamic Query Model for Sleep Study Outcomes Research

Abstract

by

Sulabh Jain

This research work presents a software solution, called PSG Query System, for sleep study clinicians and researchers as a part of an integrated effort for clinical and translational research. It provides an event-driven dynamic querying approach for patient cohort identification and presents various tools and techniques for data exploration and analysis in a user friendly manner. PSG Query System allows for composition of simple to advanced queries based on clinically relevant information for cohort identification. It allows the user to drill down into the details of patient data, polysomnogram recordings, and manually annotated events based on various electro-physiological channels. It further provides visualization of these digital channels corresponding to the various annotated events. The PSG Query System is being utilized in clinical trials, diagnosis, and treatment of patients based on clinical manifestations.
Chapter 1

BACKGROUND

Healthcare sector has seen tremendous growth in software solutions specifically designed for storing patient visit data e.g. EMR (Electronic Medical Record). A visit data can include one or more of patient’s demographic information, clinical history, physiological information, lab results, and billing information. The intention here has been clear to have an electronic management system in place which can support better management of hospital network but less emphasis has been given to the interest of clinical practitioners. The physicians and clinical researchers still suffer from the lack of access mechanism and tools which may help in establishing the correlation among patients with similar symptoms. The ultimate goal always remains to provide better diagnosis and treatment to the patient.

1.1 About PhysioMIMI

PhysioMIMI (Multi-Modality, Multi-Resource Information Integration environment) [23] is a multi-CTSA site project to design and develop technology enabled solutions and infrastructure in order to facilitate collection, management, and analysis of clinical data. This project had
Figure 1.1: PhysioMIMI Architecture

been funded by National Institute of Health (NIH) - National Centre of Research Resources (NCRR). This initiative had been proposed and funded to support collaborative clinical and translational research using diverse data types across institutions. It is build to integrate and extend existing informatics initiative called MIMI and Honest Broker.

MIMI is a web based research management system with three independent and interconnected servers called meta-server, application server, and data resource server. PhysioMIMI consists of sleep domain ontology, role based information access architecture, and honest adapter to mediate data access across institutions.
1.1.1 VISAGE

VISAGE [24] is a query interface that comes with the existing release version of PhysioMIMI. The purpose here is to create queries without knowledge of SQL language or database schema. The idea here is to give end user access to a powerful, though lightweight, set of statistical analysis tools to quickly view data distribution in queries. This allows for faster comparison of data set on a variety of axes.

1.2 Utilizing Annotations as Query Filter

Sleep study tests are stored in electronic format using EDF file format. It stores physiological channel data as well as annotated information. The extraction of annotated information from EDF files was completed in our previous research [20]. Here, we were successfully able to extract annotated information in textual format and clipped corresponding physiological channel data. This digital information was clipped purposefully to optimize performance of database queries by storing minimum information. Further, the current work discusses about the various features and strongly suggests to technologically evaluate the storage mechanism for storing digital signals for the one complete study (without clipping). And we should ideally be storing markers along study timeline to represent annotated data.

1.3 Motivation

The software system designed hereby is adaptable to the needs of various divisions across neurology such as epilepsy, strokes, neuromuscular, and so on. For example, in epilepsy department, physicians and fellows ideally want to access patient data along with EEG reports to come up with a set of patients with past history of certain focal seizures. The proposed and implemented
software system is capable of searching patient sets in an automated way. Usually a fellow within any neurology department may spend an average of 100 hours by going through a patient database consisting of approximately 10K patients. The aim is to come up with a patient cohort with past history of certain clinical findings. The person has to manage some sort of excel file with all the important findings along with back reference of the database. The proposed software system described in this research work is capable of bringing down 100 hours of efforts to 5 minutes along with an auto generated report.

In the past, there were some references for the arising need of querying mechanism for healthcare professionals but the technology was not mature enough to support those ideas. A major breakthrough came with PhysioMIMI which went on to support patient cohort identification across institutions. But, still there existed a need of a comprehensive search interface capable of supporting every patient related data including lab findings, diagnosis, and treatment. PSG Query System turns out to be a matured effort to fulfill the gap and provide comprehensive solutions in the field of medical informatics.

1.4 Current Work

Current work uses clipped digital signals along with annotated text data as seen in [20]. The idea was to integrated this annotated information as a part of search terms in query interface and further extend data exploration features along with digital signal visualization. This work presents a unique approach of seamless integration of cohort identification and signal visualization.

The research work described hereby has been conducted in collaboration with the Sleep Medicine Division to identify the technological tools that
Figure 1.2: Software Components

makes the life of clinical practitioners easier and helps in converting the weeks of effort into minutes. A need for event driven patient cohort identification framework had been established and the benefits were imminent to all.

Figure 1.2 shows the distribution of various components contained within PSG Query System. A version of Integrated Query Formulator existed previously by the name of VISAGE. This component is now added with a dedicated event driven component within its query interface. This event based search component exists along side the generic ontology based search term inclusion component. This is the reason of this particular component overlapping with PhysioMIMI. Other 2 components hanging apart are extensions of PhysioMIMI.

There is an ongoing effort to merge these new features into existing PhysioMIMI framework. This new framework will be called PhysioMIMI Hybrid. This current research focusses on the application layer on the existing architecture.
1.5 Organization of Thesis

In further chapters, a successful research work performed in developing an event driven patient cohort identification mechanism is explained further in detail. The work describes the tools for browsing patient information along with lab results in a correlated environment.

In upcoming chapters, Chapter 2 gives an overview of Sleep Medicine. Chapter 3 explains complete PSG Query System from an end user perspective. Chapter 4 gives an overview of the database design and tables that are foremost important in the design of software solution. Chapter 5, Chapter 6, and Chapter 7 explains in depth the details about the system and the working model behind each of the components within PSG Query System. Chapter 8 evaluates the current system against existing ones and evaluates the performance. Chapter 9 discusses about the conclusion and possible future directions for improvements on the top PSG Query System.

The work has been organized keeping in mind the interest of clinical practitioners, researcher, software developers, and a person interested in getting an overview of the research that lies at the intersection of neurological science and computer science.

Clinical practitioner/researcher looking for an overview of the technological environment and it’s utilization as a part of medical informatics are recommended to read chapters 2, 3, 8, and 9 in order. For a software developer interested in research in the domain of sleep medicine or neuroscience should read all the Chapters in given order. People interested in reusing and extending the research using similar technological environment should go through all the content mentioned in appendix. A person interested in getting a quick overview of the research in medical informatics is recommended to read chapter 1, 2, 3, and 9.
Chapter 2

SLEEP MEDICINE

2.1 Introduction

Sleep medicine is a medical specialty or sub-specialty devoted to the diagnosis and therapy of sleep disturbances and disorders [10]. Sleep disorders are common in all sections of the population and are either the main clinical complaint or a frequent complication of many conditions for which patients are seen in primary care or specialist services.

However, the area of Sleep Medicine was poorly covered until recently [25]. A major consequence being the manifestations of many sleep disorders now identified were likely to be misinterpreted as other clinical conditions of a physical or psychological nature, especially neurological or psychiatric disorders [25]. Sleep Medicine is now a recognized sub-specialty within anesthesiology, internal medicine, family medicine, pediatrics, otolaryngology, psychiatry, and neurology in US [10].

2.2 Polysomnography

Polysomnography, also known as sleep study, is a commonly used multi-parametric test in the study of sleep and as a diagnostic tool in sleep
medicine. It is often considered the criterion standard in the diagnosis of obstructive sleep apnea syndrome (OSAS), determining the severity of the disease, and evaluating various sleep disorders that can exist with or without OSAS. Polysomnography is a comprehensive recording of the multiple physiologic parameters related to sleep and wakefulness. It consists of multiple channels of electroencephalogram (EEG), electrooculogram (EOG), electrocardiogram (ECG), nasal and oral airflow, abdominal, chest and leg movements, and blood oxygen levels [1].

2.2.1 Components of Polysomnogram

Polysomnogram mainly consists of 3 studies: electroencephalography (EEG), electrooculography (EOG), and surface electromyography (EMG) [4].

One EEG channel (central channel with an ear reference provides the best amplitude) is used to monitor sleep stage. However, mostly 2 central channels and 2 occipital channels, with ear references as an adjunct help identify sleep latency and arousals. A 10- to 20-electrode placement system is used to determine the location of these channels. Additional EEG channels can be used, particularly in patients with epilepsy (i.e. a full 10-20 montage).

Two EOG channels are used to monitor both horizontal and vertical eye movements. Electrodes are placed at the right and left outer canthi, on above and one below the horizontal eye axis. EOG is useful is determining the onset of rapid eye movement (REM) sleep, and to note the presence of slow-rolling eye movements that usually accompany the onset of sleep.

One EMG channel (usually chin or mentalis and/or sub-mental) is used to record atonia during REM sleep or lack of atonia in patients with REM-related parasomnias. To assess bruxism, the EMG electrode can be placed over the masseter. The EMG recording from other muscle groups is assessed for other sleep disorders. Two channels are used for monitoring the airflow. One thermistor channel (oral and/or nasal) is used to evaluate the presence
or absence of airflow. Thermistor is the recommended channel for evaluation of apneas. Nasal pressure transducer channel is a more sensitive measure of airflow restriction. Pressure transducer is the recommended channel for evaluating hypopneas.

Additional parameters that can be monitored in a sleep study include the following:

1. Electrocardiography
2. Pulse oximetry
3. Respiratory effort (thoracic and abdominal)
4. End tidal or transcutaneous CO$_2$
5. Sound recording to measure snoring
6. Surface EMG to monitor limb muscles
7. Continuous video monitoring

Optional parameters that can be monitored in a sleep study include the following:

1. Core body temperature
2. Incident light intensity
3. Penile tumescence
4. Pressure and PH at various esophageal levels
2.3 Different Types of Sleep Stages

Sleep is divided into two distinct states known as Non-Rapid Eye Movement sleep (NREM) and Rapid Eye Movement sleep (REM) [3]. These two states occur in a roughly 90 minute cycle which is usually repeated 5-6 times during the night. Each cycle includes 4 stages of NREM and REM. NREM sleep is the sleep from which REM sleep emerges [3].

As we fall asleep, we enter the transition sleep called stage 1 and begin our first sleep cycle. Within a few minutes we evolve into our baseline sleep called stage 2. It is from stage 2 sleep that the other three stages emerge. Stage 2 sleep occupies approximately 50-65% of our sleep time. Within 15-20 minutes we have slowly evolved into stage 3 then stage 4 sleep, often called delta sleep or slow wave sleep because of the very high voltage, slow brain waves. Delta sleep is similar to being in a coma, but unlike a coma, it’s reversible. As we traverse these first four stages of sleep our respiration and heart rate slow and the body is almost immobile. All of a sudden, after 20-30 minutes of slow wave sleep, we lighten into stage 2 and almost immediately change gears into very active brain pattern known as paradoxical or REM sleep. Simultaneous with this dive into REM, our respiration and heart rate increases substantially and we lose our ability to use our postural or skeletal muscles.

Along with this, our brain becomes so activated that we start to hallucinate and have what we call dreams. Our eyes move down to midline, just as in wakefulness, and they begin the move sporadically, many times in relation to what we are dreaming. In effect, it’s a highly activated brain in a paralyzed body. This paradoxical state will last 10-20 minutes and then we fall back into stage 2 again. This is the end of a sleep cycle and then it starts over again, except we gradually loose our delta sleep and replaces it with longer and longer periods of alternating stage 2 and REM sleep. By
the final sleep cycle of the night, we will spend approximately half of our time in stage 2 and half in REM.

### 2.4 Sleep Disorders

Sleep disorders involve any difficulties related to sleeping, including difficulty falling or staying asleep, falling asleep at inappropriate times, excessive total sleep time, or abnormal behaviors associated with sleep.

More than 100 different disorders of sleeping and waking have been identified. They can be grouped in four main categories:

1. Insomnia

2. Excessive Daytime Sleepiness

3. Sleep Rhythm Problem

4. Sleep Disruptive Behaviors
Chapter 3

PSG QUERY SYSTEM

PSG Query System is a Patient-Cohort Identification and Analysis Software designed as a part of integrated efforts towards development of technological solutions for neuroscience department [23]. The main purpose of PSG Query System is to act as a tool for accessing and analyzing patient information stored in electronic format.

3.1 Use Cases

The typical scenarios where PSG Query System can be useful are:

- A clinical practitioner wants to come up with a set of patients from a sleep study database for an ongoing research study. This software can be used (without the help of computer professionals) for direct interaction with the patient database and identification of patient cohort satisfying clinical constraint. The software is capable of handling all the physiological parameters stored during sleep study as a part of clinical constraints.

- The software can be used as a part of academic course conducted by medical institutions. Here by, an instructor can use automated tools
3.2 Overview of the Software

PSG Query System consists of 3 main components:

1. Integrated Query Formulator
2. Patient Cohort Browser
3. Signal Visualization Tool

Each of the three components is described below along with the images that give a brief overview of the individual component. The details about the architecture and design of these components are described in Chapter 5, 6, and 7.
3.3 Integrated Query Formulator

Integrated Query Formulator is one of the most important components of PSG Query System inspired by PhysioMIMI/VISAGE [18, 27]. The selection of patient cohort is driven by Integrated Query Formulator itself. It is used in visual query formation using various physiological variables collected during the sleep study along with manually annotated events.

There are two types of selectors from which the parametric constraints can be selected for query formulation.

1. Variable selector contains all the physiological variables relevant to the diagnosis of sleep disorders. They are collected during the patient visit and are based on current/previous lab tests conducted along with demographic information and clinical history relevant to the diagnosis of sleep disorders. Examples of Physiological variables can be age, asthma condition, chronic illness, smoking habits, initial symptoms etc.

2. Event selector contains a list of various sleep study events/findings as defined in international classification of sleep disorders [26]. The list of findings is used as annotations in sleep study. These annotations include various sleep stages along with the sleep disorders. Examples of event selectors are NREM Stage 1, NREM Stage 2, REM, Hypopnea, Apnea etc.

3.3.1 Query Formulator

A selector item can be added to query interface using the above two selector panels. The query interface is further used in selection of single/multi value or a range relevant to the selector item. This is treated as an input for the current item. The selector item can be deleted from the query interface using
delete icon. As soon as the selection of input value(s)/range is performed, the query interface generated the patient count for the value constraint to that particular item. Also, it generates an aggregate patient count taking into account the complete query along with all selectors.

### 3.3.2 Logical AND/OR Operation on Selectors

A query interface can have one or multiple selector items. In case of multiple selector items, Integrated Query Formulator by default performs an AND operation on multi selection for an aggregate count. AND operation can be flipped into an OR operation using TOGGLE button.

![Figure 3.2: User Interface Displaying Toggle Button For Logical Operation](image)

![Figure 3.3: User Interface Displaying And Operator](image)
3.3.3 Rearranging Selectors using Drag and Drop

The query interface also offers a feature for rearranging selector items. This is one of the most used features in grouping/ungrouping items into different containers. Every selector panel contains a drag and drop icon which enables the user to click and drag the selectors and drop at an appropriate position. This is one of the most used features in grouping operation.

3.3.4 Grouping/Ungrouping of Selectors

Multiple Selectors and can be checked using a checkbox and grouped together in a container. To un-group, one/more selectors of the same group are checked and un-grouped using an Un-Group button in the control panel. The logical operator inside the group is always negation of the logical operator outside the group i.e. if the logical operator connecting the selector panel is AND then it will be OR inside any group. This feature has been derived from the VISAGE interface of PhysioMIMI [27].

Figure 3.4: Selector Panel Showing Drag And Drop Icon

Figure 3.5: Displaying A Group Containing 2 Selector Panels
3.4 Patient Cohort Browser

Patient Cohort Browser is a component within PSG Query System used to browse data sets of patients obtained as a result of Integrated Query Formulator. It displays the patient cohort in tabular format. There are various features available within Patient Cohort Browser which helps to preview the patient information in a quick and easy manner. Snap View feature of the browser gives detailed information about the physiological condition of the patient while the tabular format (listing patient cohort), may just give a part of it due to long list of available variables. Quick View feature displays the statistical information about the sleep study conducted for the patient. It processes the study findings and provides the distribution of sleep stages and events marked for a study. This information is displayed in the form of pie chart. Further, the patient cohort in tabular format facilitates the user to sort the cohort based on various headers available and as well as search within the cohort using a search box. There is an exporting tool available within Patient Cohort Browser which can be used to export the data in various different formats for analysis outside the application. The right most option with the tabular format can be used to get the detailed view of the patient. Detailed view is used to display the patient information along with physiological variables. It contains an embedded events/findings table which lists all the events marked and filtered during the query formulation.

3.5 Signal Visualization Tool

Signal Visualization Tool can be considered a part of Patient Cohort Browser in the current architecture but is being defined separately due to its explicit role it has within the system. Signal Visualization Tool is used to visualize different physiological parameters collected during the sleep
study. These parameters in fact correspond to various channels. Finally, an image is generated using digital signals corresponding to particular annotation and displayed on the browser. These images are generated in an optimal manner using the sampling rate fixed for a sleep study.

The database stores digital signal per channel corresponding to marked annotations. These digital signals are obtained from EDF file and stored inside database in binary format. In Patient Cohort Browser interface, there exists a list of events occurred during sleep study as explained in chapter 6. It contains a list of all the different channels available. The user can select the channel and display the digital signals as waveform on the screen.

3.6 PSG Query System - An Extension To PhysioMIMI

PhysioMIMI is a federated patient search engine to query patient set across multiple medical institutions. PSG Query System utilizes visual query interface borrowed from PhysioMIMI [8]; this helps in patient cohort identification across participating institutions. PSG Query System promoted an event driven querying and analysis software designed specifically for Sleep Study Research. PSG Query System is further said to be specialized to Sleep Study where the software is capable of evaluating manual annotations (or findings) as a search criterion. The querying interface within is driven by concepts as well as events. These concepts and events eventually map backwards to sleep domain ontology.

VISAGE [27] is useful for querying flat patient files across multiple institutions [9, 10]. While Integrated Query Formulator, built as an extension, (Query Engine within PSG Query System) is capable of performing join operation between flat patient files and manual annotations.
Since, the architecture of PhysioMIMI is built for handling federated search; network scalability can be treated as the important aspect where the application server needs to query multiple data sources. While, in PSG Query System, the focal point is the architecture built for multi table search as compared to flat file. So, query optimization is considered to be performing criterion for query interface within Integrated Query Formulator.

3.7 Redefining the Interest of Sleep Medicine Practitioners and Researchers

Medical practitioners and researcher are always looking for interesting patient cohorts which can prove to be of interest in further driving any clinical research. Traditional approach to finding a patient population of interest can be to manually look for clinical manifestations within multiple lab reports. Going through every patients report and finding the population of interest can be hard and time consuming.

A system like PSG Query System can be useful in managing the patient information along with search capabilities. It can be used in various scenarios such as a finding a patient cohort for clinical trial or for outcomes research, searching the database for any existing patient(s) with certain symptoms that may be valid to a current patient undergoing treatment. Such a system with diverse set of tools can also be used in a teaching course where the instructor and students can browse for interesting cases and visualize various channels describing an event.

PSG Query System is capable of delivering solution to a wide range of problems. The working system is place can be expanded into include similar set of tools for varying degree of data as per the future need of any institution.
Chapter 4

PSG Query System Database Modeling

Figure 4.1 represents the partial database model specifying the tables used in PSG Query System. The subsections below define the specific parts of the database and tables. This further helps in elaborate understanding of database schema used with PSG Query System.

4.1 Selection Panel

Figure 4.2 shows concepts tables which is used for storing the sleep domain ontology [6] information relevant to PSG Query System. Here, psg_variables tables contain the information about the one time information recorded for a patient during a visit. The events table stores information about the various sleep stages and classification [26].

4.2 Query Evaluation

The query formation takes place on the server side in application layer. After query formulation, the tables shown in Figure 4.3 decide on the patient cohort belonging to a pre-defined clinical manifestation. The query engine evaluates
Figure 4.1: Part Of Data Model Representing Tables Used In PSG Query System
Figure 4.2: Generation Of Selector Panels From Database Model

Figure 4.3: Query Evaluation
each nested groups using the variable information from psgstudies table and event information from events table and comes up with a patient cohort. All other groups are evaluated in a similar way. This gives patient cohort for each group. Further, union or intersection operation is performed between all the sets of patient cohort obtained from different grouping to obtain the resultant cohort.

4.3 Digital Channel Recording

Digital channel recording are stored in the different tables corresponding to each channel which are linked back to events table. For example - there is a table called ecgs which stores binary data for ECG signals corresponding to all annotations/findings stored in events tables. The database model is here de-normalized to have a separate table for each channel. This helps in case of managing sleep studies with varying recorded channels along with solving query performance issues.
Figure 4.4: Events And Associated Recording Channels
Chapter 5

INTEGRATED QUERY FORMULATOR

5.1 Introduction

Integrated Query Formulator, as the name suggests, is a query interface for all different sorts of queries relevant to finding a set of patients which satisfies clinically relevant constraints. The same query interface can be used to find patients based on physiological variables, events or combination of both. Rest of the query evaluation is performed by the query analyzer in the background.

Previously, there have been querying interfaces around where the end-user used to go through multiple web pages for selecting the search criterion and the query formulation could have been done either using variables or events. PhysioMIMI contained the querying interface where the selection criterion was driven the list of concepts within ontology framework. Integrated Query Formulator goes a step ahead where the concepts used in condition selection are categorized as variables of interest (called variables) and manual annotations (called events).
5.2 Feature Set

In Integrated Query Formulator, there are two selector panels on the left side namely variable selector and event selector. Variable selector consists of all the variables that are relevant to sleep study and can affect the outcomes research. Event selector contains the list of all sleep study findings defined in classification for sleep study disorders [26]. Right hand side of Integrated Query Formulator contains a place for query formulation. The header within query formulation consists of various available inter-logical operators between selectors namely group, un-group, and toggle.

Query formulation takes place on the right side of the panel. For every element selected using a selector widget (as shown in Figure 5.2), a panel is created inside query formulator space along with the relevant information displayed as a part of this panel. A slider widget is generated within the panel in order to select a range of values. During the value selection event, the current range of values gets displayed. On the completion of range selection, the query formulator makes an Ajax request to the database for fetching the patient count given of that particular panel. In this way, patient count is generated for every individual query panel along with their respective range of values. Also, the query engine evaluates all various operations (e.g. group, un-group) among all the panels within query interface, and makes a second Ajax request to evaluate the total count of the patients (given all the selectors are taken into account).

Figure 5.3 shows the work flow within the Integrated Query Formulator. The query interface can be used to build a query containing either only the variables or events. The underlying query engine is capable also of processing a query built using both variables and events. This is something that can be considered a pioneer feature while comparing various query interfaces that exists in medical informatics.
The header in query interface consists of 3 specific operators:

1. Group
2. Un-group
3. Toggle AND/OR

The **Group** operation is used to combine two/more different selectors into one group. On the other hand, **Un-group** operator behaves in an opposite manner by un-grouping the selectors into individual units. There exists one level of nesting for grouping the selector elements. **Toggle** (join operator) is used to toggle between AND/OR condition between selector elements. If the current outer join operation is in AND state, the nested join operation will be in OR state. When the **Toggle** button is clicked, the outer join operation flips from **AND** to **OR** while the inner join operation is toggled from **OR** to **AND**.

**Figure 5.1:** Generation Of Query Panel Using Selector Widget
5.3 Selector Widget

5.3.1 About Selector Widget

Selector Widget is a component inside Integrated Query Formulator. This subsection describes the design of widget in a generic way. The widget is specifically designed to contain large list of elements within a compact region in a web page. The widget contains a list of elements where number of elements to be visible can be customized. The list can be scrolled using a thin bar located on the right of widget. In case, the number of elements is less with all being visible in a customized height, the scroll bar tends to disappear.

5.3.2 Search Box Within Selector Widget

The widget contains a search box at the top. Since all the elements are loaded from server to client side during initialization step, search operation is performed using client side script without further interaction with the server. This helps in gaining the enhanced look and feel of the widget. During search, any input from the user is converted into regular expression and the
list is repopulated after comparison of elements with the generated regular expression.

### 5.3.3 Different Types of Selector Widgets

Figure 5.2 shows the 2 different views of the same selector widget. A. shows the widget with all the elements while the visible numbers of elements being dependent on the height of widget. B shows a state of widget while the search operation is taking place. The search operation here is instinct such the widget states gets changes as soon the user types in something.

The widget also stores the selection-state of all the elements. If any of the elements is selected to be included in query formulation, search operation preserves the selection-state of elements.

### 5.4 Design Mechanism and Workflow

The work-flow diagram as shown in Figure 5.3 represents the pictorial view of Integrated Query Formulator. This diagram covers the client-server interaction performed inside Integrated Query Formulator for patient cohort identification. The interaction can be described in the following steps:

1. When an instance of Integrated Query Formulator is generated using the web interface, the two selector widgets (variable selector and event selector) fetch the elements contained from the back-end database. This explains the dynamic behavior of selector widget such that the list of elements within variables and events can be varied based on the research outcomes study.

2. Query elements are selected from ‘variable’ and ‘events’ selector boxes. When an element is selected, a panel is created inside query formulation
Figure 5.3: Integrated Query Formulator Workflow
3. When the range of values is selected for specific search criterion, the query engine evaluates the range and sends an Ajax request. The response is in the form of patient count specific to the current panel on which the range selection is performed.

4. Every time the selection range is modified for any of the selector, another Ajax request is send to the server. This time, the query engine evaluated all the panels along with the nested grouping and AND/OR clauses to generate a SQL query. This SQL query is send along with the second Ajax request. The response obtained from the server in this case gives the patient count; given the entire search criterions are evaluated.

The purpose of Integrated Query Formulator is to select various filtering operations and retrieve the count of patients satisfying the filters. The detailed information about the patients in a cohort can be seen using Patient Cohort Browser described in next chapter.
Chapter 6

PATIENT COHORT BROWSER

6.1 Introduction

Patient Cohort Browser provides web tools for interacting with a set of patients. In other words, these tools help in drilling down into the patient information. For each patient, there is a set of associated physiological variables as well as events indicative of sleep study findings. These events are interpreted after reviewing multiple electro-physiological channels. The purpose here is to provide an intuitive browsing interface so that the whole lot of information can be reviewed by the user in an organized and convenient manner.

Patient Cohort browser has been built using Ajax technologies. This helps in integrating multiple features in an organized and intuitive manner. The web features broadly display the patient cohort in tabular format which can be further used in filtering, sorting, and rearranging the variable columns. The tabular view contains 3 different web tools i.e. Quick View, Snap View, and Detailed View for each patient in the cohort.

Patient Cohort Browser shows a list of patients spread across multiple
pages with 10 patients on each page. Multiple variables can be displayed for the patient as shown in column header. These headers can be sorted in either ascending or descending order. The search box at the top right corner of table is used to filter the rows based on the text entered.

6.2 Components of Patient Cohort Browser

6.2.1 Quick View

The tabular view contains an icon for Quick View for every patient inside the first column. Quick View is used to display statistical information about the quality of sleep in the form of dashboards. It shows two pie charts giving information about the time distribution of various findings and sleep stages. This helps in getting an overview of the events and sleep stages that occurred during the study as well as provides side-by-side comparison of the two things. When the mouse is hovered on the section of any pie-chart, it displays the total time (HH:MM:SS) of occurrence of finding/sleep stage based on the manual annotations.

6.2.2 Snap View

The icon next of Quick View is for Snap View. On click, Snap View gives the complete picture of the physiological information relevant to the patient. The purpose of Snap View is to have a complete clinical view of a patient. This consists of the patient’s age, family history, allergies, and past conditions that may be relevant to sleep study. Since the tabular view is spacious enough to gives some amount of information equally to counted as an overview of the patient in the cohort, hence, Snap View becomes a much needed functionality to serve the purpose. It is opened as a small window within the screen along with dimming the rest of the web page in background. This turns out to be helpful and efficient approach utilizing Ajax approach without leaving the
6.2.3 Detailed View

The extreme right column within the tabular view of patient cohort is a link to detailed view. Detailed view takes to individual patient page which contains the physiological information along with the sleep study performed. It gives the tabular view of all the events marked during the sleep study. The events table contains the list of events with ascending order of start time along with end-time, and the total duration in seconds. It contains the Signal Visualization Tool within the events table which opens the screen in place similar to Snap-View. Signal visualization tool and its design is discussed in Chapter 7. The sorting feature is available for events table which is similar to patients table. It can be used to sort events by start time, end time or by duration of event.
Chapter 7

SIGNAL VISUALIZATION TOOL

7.1 EDF Specification and Physiological Data

A simple digital format supporting the technical aspects of exchange and storage of polygraphic signals had been specified in European Data Format (EDF) specifications. Since then EDF specifications have become a standard in storing and exchange of physiological data in Electroencephalography and Clinical Neurophysiology. EDF+ came into existence in 2003 and was soon adapted as an extension to EDF standard. EDF is capable of storing annotations along with the additional relevant information. Text time-keeping, events, and stimuli are coded as text annotations in ‘EDF Annotations’ signals.

Sampling Rate is defined as the number of samples per unit of time (usually seconds) taken from a continuous signal to make a discrete signal.

The list shows the annotations used for storing hypnogram:

1. Sleep Stage W
2. Sleep stage 1
3. Sleep stage 2
4. Sleep stage 3
5. Sleep stage 4
6. Sleep stage R
7. Sleep stage M

The list shows the annotations used for sleep quality assessment:

1. Obstructive apnea
2. Central apnea
3. Apnea
4. Mixed apnea
5. Hypopnea
6. Limb movement
7. EEG arousal

7.2 Storing Annotation Signals in Database

A signal in an EDF format is a series of 2-byte samples, the subsequent samples representing subsequent integer values of that signal, sampled with equal time intervals. Database stores these series of 2 byte samples for each of the above types of signals corresponding to an annotation. It also stores the sampling rate along start and end time of the marked annotation.
Chapter 8

EVALUATION OF PSG QUERY SYSTEM

8.1 Software Feature Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>I2B2</th>
<th>VISAGE</th>
<th>PSG Query System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cohort Identification</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Statistical Inference</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>Ontology Driven Search</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Dedicated Event Driven Search</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>Simple-Advanced Query Formulation</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Patient Information Explorer</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>Signal Visualization Tool</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
</tbody>
</table>

8.2 Software Feature Evaluation

A study was conducted on 12 residents/fellows from department of neurology for evaluating the various features within PSG Query System. The purpose of this study was to rate the effectiveness/usefulness of various features. Various features were evaluated on the scale of 1-10 with 10 being most useful and
effectively implemented. The table below shows the average rating given to various features within PSG Query System.

**Table 8.2: Software Feature Evaluation**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Rating(1-10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cohort Identification</td>
<td>10.0</td>
</tr>
<tr>
<td>Statistical Inference</td>
<td>N/A</td>
</tr>
<tr>
<td>Ontology Driven Search</td>
<td>8.4</td>
</tr>
<tr>
<td>Dedicated Event Driven Search</td>
<td>8.2</td>
</tr>
<tr>
<td>Inference Based Search</td>
<td>N/A</td>
</tr>
<tr>
<td>Simple-Advanced Query Formulation</td>
<td>7.3</td>
</tr>
<tr>
<td>Patient Information Explorer</td>
<td>8.0</td>
</tr>
<tr>
<td>Signal Visualization Tool</td>
<td>9.6</td>
</tr>
</tbody>
</table>

The table above shows average assigned rating to various features. There were 2 features called Statistical Inference and Inference Based Search which are not within PSG Query System but the requirement of these features were suggested for future release. That is the reason of including these features but not assigning any rating to them.

## 8.3 Performance Testing

### 8.3.1 Testing Environment

All the experiments have been performed on Ubuntu 10.04 machine with 1.66 GHz Core 2 Duo processor and 3 GB RAM. The application uses Webrick as an HTTP server along with Ruby 1.8.7, Rails 3.0.3 and associated gems.

The server response time information has been calculated by Rails built-in profiler. It profiles every HTTP request and outputs the total time along with time spent inside active record code and rendering view template.

Server response time can also be seen as the time spent in completing certain controller-action. In model view controller architecture, all the event call backs are associated with controller-action on the server side. Controller
further interacts with model and generates the output view. So it becomes interesting to evaluate the server response time for controller-actions as a part of performance testing.

### 8.3.2 Query Engine Performance

Query engine within Integrated Query Formulator is responsible for processing the patients and associated events and finding the patient cohort satisfying the query constraints. Since, patient cohort identification is one of the driving interfaces for PSG Query System; it would be interesting to look at the performance of the query engine responsible for the processing.

As a part of performance testing conducted to evaluate the developed query engine, server response time has been compared to 7 different queries. The 7 different queries are selected at random ensuring the complexity increases from Q1 to Q7. The total time and the time taken by Active Record queries (which is a part of total time) have been compared for Q1-Q7.

Figure 8.1 shown above compares the 7 queries using a bar chart. The red bar represents the time taken by active record while blue bar is for the total.
server time. Here the graph shows the maximum total time of 446ms taken by Q5 and the maximum active record time comes out to be 308.3ms for Q6. The query engine within the software had undergone improvements during the development phase followed by iterative testing. The results shown above surely represent the satisfactory ones being well within 1 second.

The few contributing factor to the improvements were:

1. Indexing the primary keys within database tables.

2. Minimize the formation of active record queries during query formulation which in turn performs less number of interactions with the back-end database.

Certain precautions should be taken before the tests are performed. The query cache should be turned off at the application level as well as database level. One of the contributing steps of such a high performance may be the result of backend database placed on the same local machine. In cases, where database resides on a different machine, there might be added network latency which has not been taken into account.

8.3.3 Signal Visualization Tool Performance

Second, Signal Visualization Tool is tested for server response time. Figure 8.2 shows the comparison of average response times obtained by varying the event time duration (in seconds). X axis below shows the duration of events while Y axis shows the average server response time in milliseconds. The figure shows the performance test for ECG channel. Here, it can be easily seen that there is an approximate increase in server time. This is likely due to the size of electrical signals being directly proportional to event duration. Since the database model in unbiased towards any channels, varying the channels will always produce a similar
pattern. One point to consider is the sampling rate factor. For the channels with greater sampling rate, the size of digital data contained in database is greater for the channel keeping event duration fixed. That means, large chunk of binary data is to transferred for channels with greater sampling rate.

8.3.4 Task Based Approach to System Performance Evaluation

Finally, we focus on task based approach to evaluate and compare the performance of various tasks across the software. Here, the software definition of a task can be defined as all events triggered to obtain any information of relevance from the system. These can include button clicks, page reload, or option selection. Here we focus on most complicated ones that contain a ton of features and are the basis of PSG Query System.

A list of task was identified for response time comparison as explained below:

1. PSG Search Page - This task includes the loading of search filters and construction of user interface for query builder.
2. Patient Cohort Identification - This is one of the complex features within the software and elaborate study later. It explains the response time on each filter selection along with range of values. Since various different queries can be constructed, so figure below takes an average response time after an exhaustive experimentation.

3. Patient Cohort View - It is the time taken for loading the patient cohort browser given the system has already identified a patient cohort.

4. Quick View - It indicated the visualization time of quick view panel including dashboards which give the statistical information about the patient study.

5. Detailed view - It is the response time of the detailed view which takes the user to detailed information about the patient along with all event information.

6. Signal Visualization - It is an average response time taken by image visualization tool. An elaborate study conducted for this feature in discussed in Section 9.2.

Figure 8.3 shows the average server response time for comparison of task based events described above. Here we can see the response time for the entire set of tasks within 500ms with detailed view taking the maximum time out of all.
Figure 8.3: Server Response Time For Various Tasks
Chapter 9

CONCLUSION AND FUTURE WORK

9.1 Conclusion

The research work presents here an innovative querying model for the needs of sleep study clinicians. This work had been performed in given different stages:

- Identify the needs to clinicians and researchers who frequently need to search for patient set based on clinical constraints.

- Organize an estimated set of features in parallel with the routine workflow.

- Design a query mechanism to include patient specific data along with sleep study tests conducted.

PSG Query System is a software solution in align with the above requirements of physicians, fellows, and course instructors. It demonstrates various different features that can ease the life of these people in performing their daily tasks as well as accelerate the research outcomes to the achieve
highest level of standards. The purpose of the research was not to provide a comprehensive set of features but can be laid down as follows:

- First, to architect and design a patient data querying cum browsing model which aligns with the need to clinicians.

- Second is to present different possible features that can be integrated into such system. This helps us to realize the potential of technology that can be integrated with patient data in providing database search solutions.

- Finally, organize the set of features in align with the need and routine workflow.

9.2 Future Work

This section discusses about the future work from research perspective as well as software developer perspective. Understanding the future from 2 different viewpoints surely broadens the domain knowledge and utilization computer science in the field of medicine.

There is always a scope of endless technological innovation in the field of medicine and we would rather focus of the immediate requirements given we already have a working PSG search mechanism in place. The foremost task that remains is the seamless integration of domain information within database model with sleep domain ontology. This should be in a direction such that the data models become more independent of domain knowledge. Domain independent data model integrated with dynamic data dictionary finds the most of its use across disciplines in the field of medicine.

Since PhysioMIMI is more towards federated querying interface while PSG Query System supports event oriented single-institutional query interface along with broad range of tools for browsing associated
information. There are 3 technological visions with increasing complexity that can be considered as extension at the intersection of PSG Query System and PhysioMIMI. First, there should be framework for continuous evolution of domain ontology which engages the medical domain experts for discussion and standardization of domain ontology along with associated relation. Second, the data model of PSG Query System/PhysioMIMI should be made as much independent of domain knowledge as possible. This will set the path of generic multi-institution patient cohort identification and browsing associated features as shown in PSG Query System. The better generalization of such a system will lead to a wider adaptation of the combined system. Third, the system should be capable enough of organizing clinical/research outcomes study; a physician should be able to organize a research study based on the existing ontology for a fixed term and further be able to reuse the data analysis features provided in PhysioMIMI and PSG Query System on an ongoing basis.
Appendix A

Software Development Environment

PSG Query System has been developed using Ruby on Rails web application framework. It uses Ruby v1.8.7 along with Rails v3.0. Section B gives an overview of Ruby on Rails framework along with references which might be useful in learning Ruby on Rails development.

Most of the client side script has been developed using jQuery, jQuery UI and various supported plugins. Although most of the client side features use javascript and jQuery, PSG Query still has support for Prototype library. This support for prototype library has been maintained, the reason being PhysioMIMI uses prototype as the client side library for all its features.

It is highly recommended to use Unix based platform for Ruby on Rails development. Various distributions of Linux (e.g. Ubuntu, Fedora) and Macintosh have been supported well in development efforts. The database is hosted using MySQL v5.1. The code is hosted and managed using Git server. Each developer in a team uses a local Git repository and pushes the changes to the shared Git repository/server.
Appendix B

Ruby on Rails

Ruby on Rails is an open source web application framework for the Ruby programming language. It uses the Model-View-Controller (MVC) architecture pattern to organize application programming. It has easy to incorporate mechanisms for multiple javascript libraries like Prototype, Script.aculo.us, jQuery for Ajax. It uses RESTful software architecture for developing web services within the web application.

Ruby on Rails is intended to emphasize Convention over Configuration, and rapid development principle of Don’t Repeat Yourself (DRY).

Ruby on Rails application relies on a web server to run it. Passenger, one of the recent web server for Ruby on Rails, is finding wide acceptance among the Rails community. Other web servers that are commonly used for running a application include Mongrel, WEBrick etc.

B.0.1 Recommended Reading


2. The Ruby Way (2nd Edition)

3. Ruby Cookbook
4. Agile Web Development with Rails (4\textsuperscript{th} Edition)

5. The Rails 3 Way (2\textsuperscript{nd} Edition)
Appendix C

Example Code

C.1 Event Selector Widget

Ajax Request From View

```javascript
/* function to generate event panel */
$.fn.event_panel = function(){
    var response = $.ajax({
        type: "POST",
        async: false,
        dataType: "json",
        url: "/events/getEventList",
        success: function(response) { }
    });
    var events = jQuery.parseJSON(response.responseText);

    $(this).append("<div>Event Type:<select>
    <option value='0'>All</option>
    </select></div>");

    for(i = 0; i < events.length; i++) {
        $(this).find("select")
            .append("<option value='" + events[i].term.id +"'>" +
                events[i].term.name + "</option>");
    }

    $(this).find("div").css({'border': '2px solid #D3D6FF', 'padding': '10px'});
    $(this).find("select").selectBox();
    $(this).find(".selectBox").css({'width': '240px'});
};
```
Controller Action To Retrieve and Return Events

```ruby
# Method to retrieve event list
# POST REQUEST
def getEventList
    # get all events
    @terms = Term.all
    respond_to do |format|
        format.json { render :json => @terms.to_json }
    end
end
```

C.2 Snap View Generator

```javascript
(function($) {

    $.fn.addSnapViewer = function() {
        // create a dialog box element
        var $dialog = $(
            "<div id='dialog'><p>Patient Details</p></div>"
        );
        $dialog.dialog({
            autoOpen: false,
            show:"blind",
            hide:"explode",
            height: 300,
            width: 500,
            modal: true
        });

        var $stableHandle = $(this).find("#patient_list");

        // Click Event for Patient Row
        var $pRow = $stableHandle.obtainDialogHandle();

        // disable row click
        $pRow.click( function() {
            $pat_id = $(this).find("td:first");
            $("#dialog").dialog("open");
            return false;
        });
    };

    $(this).append($dialog);
})(jQuery);
```
C.3 Signal Visualization

This section shows an example code on retrieving digital signals stored in database and displaying on the user screen.

C.3.1 Ajax Request Initiated From Client

```javascript
/* function to generate signal visualization
 * params:
 * event_id - event/annotation id
 * container - html tag container where signal visualization is displayed
 * channel_name - name of the digital channel
 */
function displaySignals(event_id, container, channel_name) {

    /* read binary data string and convert into integer array */
    var signal = jQuery.ajax({
        type: 'GET',
        async: false,
        data: {event_id: event_id, channel_name: channel_name},
        dataType: 'json',
        url: '/home/getSignalValuesByPsgId'
    });

    /* read graph options */
    var graph_options = jQuery.parseJSON( jQuery.ajax({
        type: 'GET',
        async: false,
        data: {event_id: event_id, channel_name: channel_name},
        dataType: 'json',
        url: '/home/getChannelArgs'
    }).responseText);

    // store graph options in an object
    var options = [];
    options.sampling_rate = graph_options.rate;
    options.start_time = graph_options.start_time;
    options.graph_title = graph_options.graph_title;
    options.series_name = graph_options.series_name;
    options.x_title = graph_options.x_title;
    options.y_interval = graph_options.y_interval;
    options.y_title = graph_options.y_title;

    // function called to display graph
    displayGraph(container, signal.responseText, options);
}
```
C.3.2 Controller Action Code On Server Side

```ruby
# GET Request
def getSignalValuesByPsgId
  # extract url parameters
  @event_id = params['event_id']
  @channel_name = params['channel_name']

  # get model name from channel name
  @model_name = @channel_name.classify

  # apply correction due to extra s missing in model name
  if @model_name.to_s == 'ThorRe' then
    @model_name = 'ThorRes'.to_s
  elsif @model_name.to_s == 'AbdoRe' then
    @model_name = 'AbdoRes'.to_s
  end

  # get digital signal in binary format
  @sr = @model_name.constantize.find(:first, {}
    :conditions => 'eventID=#{@event_id}').signalValues;

  # convert digital signal from binary format to integer array
  count = 0
  arr = []
  @i = 0
  while @i < @sr.size
    # 2 byte sample represents one integer as per EDF specs
    # s = 16 bit signed integer, l = 32 bit signed integer
    arr[count] = @sr[@i..@i+1].unpack('s')[0]
    count += 1
    @i += 2
  end

  # send integer array to client side in json format
  respond_to do |format|
    format.json { render :json => arr }
  end
end
```

C.3.3 Graph Generation Using Highchart Library

```ruby
// Displaying Graph
// function to display graph using highchart library
// ------------------------------

var chart;
```

function displayGraph(container, responseText, options) {

    // set options object for customizing graphical display
    options.graph_title = options.graph_title || "Graph title here...";
    options.series_name = options.series_name || "Series name here...";
    options.sampling_rate = options.sampling_rate || 125;
    options.x_title = options.x_title || "x axis title...";
    options.y_title = options.y_title || "y axis title...";
    options.y_interval = options.y_interval || 200;
    options.start_time = options.start_time || "2010-01-01 00:00:00";

    // Chart Attributes
    var GRAPH_CONTAINER = container;
    var BINARY_DATA = responseText;

    var START_DATE = options.start_time; //"2001-07-18 20:32:56";
    var SAMPLING_RATE = parseInt(options.sampling_rate);
    var GRAPH_TITLE = options.graph_title; //"EKG Demo Graph";
    var SERIES_NAME = options.series_name;
    var X_TITLE = options.x_title; //"Time (in seconds)";
    var Y_TITLE = options.y_title; //"Volt (in microsecond)";
    var Y_TICKINTERVAL = options.y_interval;

    // DERIVED ATTRIBUTES
    var YYY = parseInt(START_DATE.substr(0,4));
    var MM = parseInt(START_DATE.substr(5,2)) - 1;
    var DD = parseInt(START_DATE.substr(8,2));
    var HH = parseInt(START_DATE.substr(11,2));
    var MIN = parseInt(START_DATE.substr(14,2));
    var SEC = parseInt(START_DATE.substr(17,2));
    var MSEC = 0;

    // highchart function to generate chart
    // refer to highcharts library for additional information
    chart = new Highcharts.Chart({
        chart: { renderTo: GRAPH_CONTAINER, 
            defaultSeriesType: 'spline', 
            zoomType: 'x' 
        },
        credits: { enabled: false },
        title: { text: GRAPH_TITLE },
        xAxis: { lineWidth: 2, // width of axis line 
            title: { text: X_TITLE},
            type: 'datetime',
            dateTimeLabelFormats: { second: ':%H:%M:%S' } 
        },
        yAxis: { lineWidth: 2, // width of axis line 
            tickInterval: Y_TICKINTERVAL,
            title: { text: Y_TITLE } 
        },
        plotOptions: { spline: { 
            lineWidth: 1,
            states: { hover: { lineWidth: 1 }},
            marker: { enabled: false }
        } },
        series: [{
            name: SERIES_NAME,
            data: jQuery.parseJSON(BINARY_DATA),
            pointStart: Date.UTC(YYY, MM, DD, HH, MIN, SEC, MSEC),
        }]
    });
}
enableMouseTracking: false,
// one sec / 250 sampling size = 4ms point interval
pointInterval: (1000 / SAMPLING_RATE)
}}
}};
}
Bibliography


[22] NIH-NCRR. Sleep, sleep disorders, and biological rhythms. Online.


