ONTOGRAPHY-BASED, INTERFACE-DRIVEN DEVELOPMENT OF CLINICAL DATA MANAGEMENT SYSTEMS

by

SHIQIANG TAO

Submitted in partial fulfillment of the requirements For the degree of Doctor of Philosophy

Dissertation Advisor: Dr. Guo-Qiang Zhang

Department of Electrical Engineering and Computer Science

CASE WESTERN RESERVE UNIVERSITY

May 2016
CASE WESTERN RESERVE UNIVERSITY
SCHOOL OF GRADUATE STUDIES

We hereby approve the thesis/dissertation of

Shiqiang Tao

candidate for the Ph. D. degree*. 

(signed) Dr. Guo-Qiang Zhang, Ph.D. 

(chair of the committee)

Dr. Kenneth A. Loparo, Ph.D.

Dr. Xiang Zhang, Ph.D.

Dr. Samden Lhatoo, Ph.D.

Dr. Licong Cui, PhD.

(date) December 1, 2015

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

1 Introduction ........................................... 1
   1.1 Challenges in Clinical Data Management ............... 1
       1.1.1 Identify Eligible Patients for Clinical Trials .......... 2
       1.1.2 Effective Capture of Multi-modality Clinical Data ....... 2
       1.1.3 Patient Cohort Identification in Complex Clinical Procedure 3
   1.2 Solutions and Contributions .......................... 4
   1.3 Organization of the Thesis ................................ 6

2 Background ............................................ 7
   2.1 Trial Prospector ...................................... 7
       2.1.1 Clinical Trials .................................. 7
       2.1.2 Eligibility Criteria .............................. 8
       2.1.3 Challenges of Subject Recruitment in Clinical Trials .... 9
       2.1.4 Informatics-based Approaches ...................... 9
   2.2 OPIC: Ontology-Driven Patient Information Capturing System for
       Epilepsy ............................................ 10
       2.2.1 Sudden Unexpected Death in Epilepsy (SUDEP) ........... 10
       2.2.2 The Center for SUDEP Research (CSR) .................. 10
       2.2.3 Epilepsy Center of UHCMC ........................ 11
   2.3 DBSDB: Database for Deep Brain Stimulation ............. 12
2.3.1 Parkinson’s Disease ........................................... 12
2.3.2 Deep Brain Stimulation ...................................... 12

3 Configurable Role-Based Access Control ........................................... 14

3.1 Background ........................................................................ 15
  3.1.1 Access Control ................................................................ 15
  3.1.2 Access Control Models ..................................................... 16
  3.1.3 Role Based Access Control ............................................... 16

3.2 Design and Implementation ..................................................... 18
  3.2.1 Privilege Assignment Table .............................................. 20
  3.2.2 Role Assignment Table ..................................................... 20
  3.2.3 Three Access Control Mechanisms ..................................... 21

3.3 Results .............................................................................. 23
  3.3.1 Role-based Access Control in Trial Prospector .................... 23
  3.3.2 Role-Based Access Control in Ontology-Driven Patient Infor-
    mation Capturing System for Epilepsy ..................................... 24
  3.3.3 Role-based Access Control in Database for Deep Brain Stim-
    ulation ............................................................................ 26

3.4 Discussion ......................................................................... 28

4 Ontology Guided Data Capture ..................................................... 31

4.1 Background ........................................................................ 32
  4.1.1 The Center for SUDEP Research (CSR) ............................ 32
  4.1.2 Role of OPIC ................................................................. 33

4.2 Method ............................................................................. 33
  4.2.1 Use Cases ...................................................................... 33
  4.2.2 OPIC Report Structure .................................................. 35
4.2.3 Ontology Guided Data Capture in Classification of Paroxysmal Episodes ................................. 38

4.3 Results .......................................................................................................................... 45

4.4 Conclusion and Discussion ........................................................................................ 46

5 VISAGET – Visual Aggregator and Explorer for Clinical Trials ............................................ 47

5.1 Background .................................................................................................................. 48

5.1.1 Common Data Elements (CDEs) ............................................................................. 48

5.1.2 VISual AGgregator and Explorer (VISAGE) ....................................................... 49

5.2 Methods ........................................................................................................................ 49

5.3 Eligibility Criteria Ontology .......................................................................................... 50

5.3.1 Attributes of ECO ............................................................................................... 50

5.3.2 Build Eligibility Criteria from ECO ...................................................................... 52

5.3.3 Modeling of Clinical Trials .................................................................................... 54

5.3.4 Patients in VISAGET ............................................................................................ 55

5.3.5 Matching Patients with Clinical Trials .................................................................... 56

5.4 Implementation of VISAGET ....................................................................................... 57

5.4.1 Build Eligibility Criteria Ontology ........................................................................ 57

5.4.2 Build ECO from Common Data Elements ............................................................. 58

5.4.3 Coverage of Semantic Classes for Eligibility Criteria ....................................... 58

5.4.4 Create or Edit from System Interface .................................................................... 60

5.4.5 Interactive Interface for Building Eligibility Criteria ......................................... 61

5.4.6 Matching Algorithm .............................................................................................. 64

5.5 Pilot Results of VISAGET ........................................................................................... 66

5.5.1 Patient Eligibility ................................................................................................. 66

5.5.2 User Evaluation ..................................................................................................... 67

5.5.3 Performance Scalability Evaluation ....................................................................... 68

5.6 Conclusion and Discussion ........................................................................................... 69
6  Active Dashboard

6.1  Background ................................................. 71
   6.1.1  Parkinson’s Disease Treatment .................... 71
   6.1.2  Deep Brain Stimulation Workflow .................. 72
6.2  Methods .................................................... 73
   6.2.1  Data Form ............................................ 73
   6.2.2  Clinical Visit ....................................... 74
   6.2.3  Patient State ...................................... 75
   6.2.4  Patient State Transfer ............................. 76
   6.2.5  Architecture of Active Dashboard .................. 77
   6.2.6  Best Practices in Database for Deep Brain Stimulation .... 77
6.3  Results ..................................................... 80
   6.3.1  Patient States Library ............................. 80
   6.3.2  Active Dashboard User Interface ................... 80
6.4  Evaluation ............................................... 82
6.5  Conclusion and Discussion .............................. 84

7  Web Interface Driven Development .................... 85

7.1  Motivation ............................................... 85
7.2  Roles and Responsibilities .............................. 87
   7.2.1  Project Manager .................................... 87
   7.2.2  Chief Architect .................................... 88
   7.2.3  Lead Developer .................................... 88
   7.2.4  Developers .......................................... 89
   7.2.5  Domain Experts .................................... 89
   7.2.6  Meeting Coordinator ............................... 89
7.3  Processes ............................................... 90
   7.3.1  Build Interfaces List .............................. 90
7.3.2 Plan by Interface ........................................... 91
7.3.3 Interface Lifecycle ........................................ 91
7.4 Interface Lifecycle ........................................... 91
7.4.1 Requirements Details Collection ......................... 91
7.4.2 Initial Interface Building ................................. 93
7.4.3 Review Session ............................................ 94
7.4.4 Interface Revision .......................................... 94
7.4.5 Implementation of Model and Controller ............... 95
7.5 Practice .......................................................... 95
7.5.1 Requirements Details Collection ......................... 95
7.5.2 Initial Interface Building ................................. 96
7.5.3 First Review Session ....................................... 97
7.5.4 First Revision .............................................. 98
7.5.5 Second Review Session .................................... 100
7.5.6 Second Revision ........................................... 101
7.5.7 Third Review Session ...................................... 102
7.5.8 Implementation of Model and Controller ............... 102
7.6 Development Environment ................................. 102
7.6.1 Ruby on Rails .............................................. 102
7.6.2 jQuery and jQuery UI .................................... 103
7.6.3 Bootstrap .................................................. 103
7.6.4 MySQL ..................................................... 103
7.7 Results .......................................................... 103
7.8 Conclusion and Discussion ................................. 104
7.8.1 Applicability ............................................... 105

8 Conclusion ....................................................... 106
8.1 Deployment in Production Systems ......................... 108
8.2 Future Work ......................................................... 109

8.2.1 Improve Proposed Methods ................................. 109

8.2.2 Propose New Methods ........................................ 109

Appendices

A Epileptic Seizure Terminologies ................................. 110

B Source Code in Ruby On Rails ................................. 112

B.1 Access Control Mechanisms of cRBAC ....................... 112

B.2 Implementation of Tider in Ontology guided Data Capture .. 115

B.3 Matching algorithm for VISAGET .............................. 118

B.4 Active Dashboard ............................................... 128

B.4.1 Interface Development ...................................... 128

B.4.2 The Query Algorithm ...................................... 133
List of Tables

3.1 Critical fields in concepts .............................................. 19

4.1 All nine categories of modifiers in OPIC .......................... 40

5.1 Build single eligibility criterion from eligibility criteria ontology ... 53
5.2 Build group eligibility criterion from eligibility criteria ontology ... 54
5.3 Mappings between value types and visualization widgets .......... 62

6.1 Patient states of DBSDB .................................................... 81

7.1 VISAGET development environment ............................... 102
7.2 Distinct forms in DBSDB and the number of data fields captured in them 104
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Architecture overview of clinical data management system</td>
<td>4</td>
</tr>
<tr>
<td>3.1</td>
<td>RBAC module is located after the authentication and before any operations can be done.</td>
<td>18</td>
</tr>
<tr>
<td>3.2</td>
<td>Concept relationships in configurable role based access control</td>
<td>19</td>
</tr>
<tr>
<td>3.3</td>
<td>Relation table that assigns privileges to roles</td>
<td>20</td>
</tr>
<tr>
<td>3.4</td>
<td>Relation table to assign roles in cores to a user</td>
<td>21</td>
</tr>
<tr>
<td>3.5</td>
<td>Interface for privilege assignment and role assignment in Trial Prospector</td>
<td>25</td>
</tr>
<tr>
<td>3.6</td>
<td>Interface for privilege assignment and role assignment in OPIC</td>
<td>27</td>
</tr>
<tr>
<td>3.7</td>
<td>Interface for privilege assignment and role assignment in DBSDB</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>Use cases in OPIC</td>
<td>34</td>
</tr>
<tr>
<td>4.2</td>
<td>OPIC report structure: 12 independent data sections and one operation panel</td>
<td>36</td>
</tr>
<tr>
<td>4.3</td>
<td>Modeling of epileptic seizure terminologies in episodes classification section</td>
<td>39</td>
</tr>
<tr>
<td>4.4</td>
<td>Modifier of category 2 working in episodes classification</td>
<td>41</td>
</tr>
<tr>
<td>4.5</td>
<td>Collection of 4 Tiders for OPIC Terminologies</td>
<td>43</td>
</tr>
<tr>
<td>4.6</td>
<td>New form for Episodes Classification</td>
<td>44</td>
</tr>
<tr>
<td>4.7</td>
<td>One example entry of classification of paroxysmal episodes</td>
<td>45</td>
</tr>
<tr>
<td>5.1</td>
<td>ECO representation of eligibility criteria for white blood cell count</td>
<td>52</td>
</tr>
<tr>
<td>5.2</td>
<td>Trial in VISAGET has many eligibility criteria built from ECO Term</td>
<td>54</td>
</tr>
<tr>
<td>5.3</td>
<td>Patient model in VISAGET</td>
<td>55</td>
</tr>
</tbody>
</table>
5.4 Match patient data to a clinical trial ........................................ 56
5.5 Node syntax in XML file for ECO ........................................... 57
5.6 Mappings between protocol description, ECO term, and visualization widget .......................................................... 60
5.7 System interface to create ECO term ...................................... 61
5.8 Interactive trial builder ......................................................... 62
5.9 Matching Algorithm ............................................................ 65
5.10 Distribution of exclusion reasons ......................................... 67
5.11 Execution times to match different size patients to different size of clinical trials .......................................................... 69

6.1 Illustrative diagram for the DBS workflow in UH movement disorders center. ................................................................. 73
6.2 One clinical visit may generate many data forms ....................... 74
6.3 Relationship between patient and patient states: one patient may have many patient states .................................................. 75
6.4 State transfers of a patient state ............................................. 76
6.5 Architecture of active dashboard .......................................... 78
6.6 Diagnoses selector in DBSDB ............................................... 79
6.7 Query interface of active dashboard ...................................... 81
6.8 Statistics about patient statuses captured in DB2S .................... 83

7.1 Three processes of web interface driven development. ............... 90
7.2 Five steps of interface implementation life cycle in WIDD. .......... 92
7.3 Initial data modeling related to programming interface .............. 96
7.4 Initial building of overview page in programming interface ....... 97
7.5 Initial building of edit page of a programming record .............. 97
7.6 Revised data modeling related to programming interface .......... 99
7.7 Revised of overview page in programming interface ................. 99
7.8 Revised edit page of a programming record ......................... 100
7.9 Benefit structure ................................................... 100
7.10 Side effect structure ................................................ 101
7.11 Revised programming session page after the second review session .... 101
Acknowledgements

This thesis represents not only my work on the keyboard, but also a milestone in more than five years of work at Case Western Reserve University (CWRU). My experience at CWRU has been amazing since my first day on August 3rd, 2010.

First and foremost I wish to thank my advisor, Professor GQ Zhang. Thanks to him, I have the opportunity to pursue my PhD degree at CWRU. He has been very supportive since the first day I began my study in CWRU. Dr. Zhang’s profound insight, constant support and guidance have been essential through my entire PhD study. This dissertation would not be possible without his mentorship.

Besides my advisor, I would like to thank my PhD dissertation committee members, Dr. Kenneth Loparo, Dr. Xiang Zhang, Dr. Samden Lhatoo, and Dr. Licong Cui, for their time and valuable feedback that have helped improve my dissertation work.

I would like to thank the members of Dr. Zhang’s group for their help and friendship. They include Sridevi Cherukuri, Jeralynne Androsko, Jeno Mozes, Dr. Lingyun Luo, Dr. Zhihui Luo, Van Anh Tran, Catherine Jayapandian, Yongbo Niu, Yang Chen, Wei Zhu, Meng Zhao, Sisi Gu, Tarun Jain, Yash Gunapati, Pei Zhao, WeiWang, Priya Ramesh, Mengmeng Sun, Ningzhou Zeng, Xi Wu, and Xiaojin Li.

Last but not least, I would like to thank my mom for her unconditional love and support for my education. I would like to thank my wife for her understanding and help during my dissertation writing. I would also thank my baby daughter Miyue. She is such a happy baby that her smiles help me stay positive and motivated.
List of Abbreviations

- CDMS: Clinical Data Management System
- CDE: Common Data Element
- CMC: Case Medical Center
- DAC: Discretionary Access Control
- EHR: Electronic Health Record
- EMR: Electronic Medical Record
- EMU: Epilepsy Monitoring Unit
- FCA: Formal Concept Analysis
- IS: Information System
- LBAC: Lattice-based Access Control
- MAC: Mandatory Access Control
- MRN: Medical Record Number
- SNOMED CT: Systematized Nomenclature of Medicine Clinical Terms
- SUDEP: Sudden Unexpected Death in Epilepsy
- UH: University Hospitals
- VISAGE: VISual AGgregator and Explorer
- VISAGET: VISual AGgregator and Explorer for Clinical Trials
Ontology-Based, Interface-Driven Development Of Clinical Data Management Systems

Abstract

by

SHIQIANG TAO

Although electronic health record (EHR) has been used for decades and progress has been made in its adoption, paper or document based forms are still in wide use in various clinical settings such as epilepsy center, movement disorder program, and cancer center. Effective data management approaches and systems are needed to be able to make use of the data generated from patient care in order to improve outcomes.

In this thesis, we present the design, implementation, and evaluation results from three live clinical web systems: Trial Prospector, Ontology-driven Patient Information Capture system for epilepsy (OPIC), and DataBase system for Deep Brain Stimulation (DBSDB). These systems are for different clinical programs and have different application purposes, but the challenges faced in these programs are common and the methods and solutions introduced for these systems are general and interrelated. These methods work together to effectively complete clinical data management tasks, such as data capture, navigation, query, and visualization.

The contributions of the thesis consist of: a generic configurable role-based access control management module to systematically authorize different group of users to the system according to their specific responsibilities; a design methodology of ontology-guided data capture; an ontology-driven interactive interface to build comprehensive queries for patient cohort identification; a new approach for data
organization using a technique called active dashboard; and an improved method for agile software development.

Trial Prospector manages 85 active clinical trials and has matched more than 5,374 patients to these trials to find eligible participants; OPIC has generated 1140 discharge summaries; DBSDB has captured more than 1,000 clinical data forms for 264 patients. DBSDB has 16 active users, Trial Prospector has 68 active users, and OPIC has active 100 users. Feedback from these users indicate that our systems are robust and user friendly.
Chapter 1

Introduction

1.1 Challenges in Clinical Data Management

Clinical data are generated during the course of patient care. Clinical data includes but is not limited to patient’s demographic information, diagnosis, treatment, medication, laboratory tests, and hospitalization information. Tasks involved in clinical data management include patient data capture and patient data retrieval. Many clinical settings face challenges in effectively managing clinical data, including:

- Paper-based or document-based patient data capture. There is no supporting electronic data management system in the clinical setting. Patient data is manually collected on paper or on electronic documents, e.g., Microsoft Word.

- Unstructured patient data. Patient data is captured electronically but stored as free text.

- Manual review of patient data. There is inefficient support for patient data query in existing data management system. Work like patient cohort iden-
ttification to find a group of patients sharing certain common characteristics has to be manually done by clinical professionals.

This dissertation focuses on addressing these challenges to help clinical settings capture well-structured data and automate patient data retrieval.

The approaches proposed in the dissertation are created from the collaboration with three clinical settings of University Hospitals Case Medical Center (UHCMC): Seidman Cancer Center, Epilepsy Center, Deep Brain Stimulation Program. We highlight specific data management needs in the three clinical settings next.

1.1.1 Identify Eligible Patients for Clinical Trials

Clinical Trials (CTs) play a critical role in translating fundamental clinical research to patient care, especially for cancer treatment. There are hundreds of active clinical trials running in Seidman Cancer Center of UHCMC. To enroll patients to a clinical trial, physicians have to manually compare information of every patient with the inclusion and exclusion criteria of the clinical trial and determine each patient’s eligibility for enrollment. Patient data is stored in different EHR systems. Protocols of clinical trials that contain inclusion and exclusion criteria are unstructured text files. The manual process is very time consuming and physicians increasingly complain about lack of time. This has been reported [1] as an important practical barrier for patient recruitment of clinical trials.

1.1.2 Effective Capture of Multi-modality Clinical Data

For every discharged patient, the Epilepsy Center of UHCMC needs to generate a discharge summary to record the clinical care that has been provided. The summary covers information such as demographic, history, medication, seizure events, evaluations, conclusions and recommendations. The discharge summaries are im-
portant for both clinical and research purposes. Before we develop and deploy our clinical data management system in the center, these discharge summaries are created and stored in word documents. This document-based data management makes it difficult for analysis and secondary use of patient data. Further, the discharge summary includes multi-modal information like text and signal data corresponding to electroencephalography (EEG), electrocardiography (EKG), and magnetic resonance imaging (MRI) evaluations, which requires tedious manual efforts from clinical experts to copy and paste into the shared word document.

1.1.3 Patient Cohort Identification in Complex Clinical Procedure

Program of Deep Brain Stimulation in UHCMC provides treatment for Parkinson’s disease. Parkinson’s disease (PD) [2] is a progressive neurological disorder characterized by movement impairments such as tremor, slowness of movement, rigidity or stiffness of the limbs and trunk, and impaired balance and coordination. Over the last two decades, Deep Brain Stimulation (DBS) [3] has emerged as a powerful clinical therapy for the treatment of PD. The success of DBS therapy relies on the implantation of micro-electrodes, and the optimal stimulation parameter settings of the electrodes. With hundreds of patients taken DBS therapies in the program of Deep Brain Stimulation of UHCMC, records generated from these patients should have been very helpful to stimulation parameters tuning for future patients. However, the existing EHR systems store these records in a unstructured way, which makes the secondary use of data difficult. Besides, DBS therapy is not appropriate for every PD patient. There are many clinical visits and evaluations before a patient can finally take the surgery, which is a complex long workflow. The existing EHR system provides no support for patient cohort search based on their clinical information. It is a labor intensive task for clinical staff to identify the patient cohort for the next step of deep brain stimulation workflow.
1.2 Solutions and Contributions

To address these challenges, we propose a general architecture called ontology-based clinical data management system (CDMS) illustrated in Figure 1.1. CDMS adopts clinical setting’s local authentication system to perform user authentication so that users don’t need to keep additional credentials; After users are successfully authenticated, CDMS authorizes users to perform operations using a role-based access control module; Authentications, authorizations, and operations are all audited in CDMS.

Our contributions are: 1) we create a generic configurable role-based access control module that can be configured to satisfy different level needs of access control. There are two kinds of operations users can perform after authorization is given: patient data capture and patient data retrieval. 2) For data capture, CDMS leverages a methodology of ontology guided data capture. Ontology acts as knowledge library to provide standard vocabularies to ensure consistency and accuracy of data capture. With the help of specially designed widgets, CDMS can
improve users’ work efficiency and avoid manual errors like typos. For patient data retrieval, two mechanisms are created in CDMS. 3) One mechanism is called VISAGET, an ontology driven interactive interface, with which users can build comprehensive queries to identify patient cohorts based on their demographic information as well as clinical information. 4) The other mechanism is called Active Dashboard. Active Dashboard creates a novel way of data organization. With this data organization, users can conveniently find target patient cohorts with a few clicks and navigate to data elements directly. 5) We create a novel agile software development methodology called web interface driven development (WIDD) for CDMS. WIDD helps us to achieve high efficiency of design, development, and deployment of clinical data management systems.

We have applied CDMS design and WIDD methodology into the creation of three live clinical systems that are now working in production environments:

- Trial Prospector – Trial Prospector is a tool developed for UH Seidman Cancer Center to automate the matching between patients and clinical trials;

- Ontology-driven Patient Information Capturing system for Epilepsy (OPIC) – OPIC is created for UH Epilepsy Center to create and manage discharge summaries;

- Database system for Deep Brain Stimulation (DBSDB) – DBSDB is designed and implemented for UH Deep Brain Stimulation program to manage their workflow and data generated from it.

Feedback from users shows these systems are robust, user friendly and solve their data management challenges.
1.3 Organization of the Thesis

We organize our dissertation as follows:

In Chapter 2, we talk about background information about the work in this dissertation.

In Chapter 3, we present our design and implementation of a generic configurable Role-based Access Control (cRBAC). There are three access control mechanisms created out of our cRBAC. We also talk about the applicable situations of these three mechanisms.

In Chapter 4, We talk about ontology guided data capture. We use one data section (Classification of Paroxysmal Episodes) of patient discharge summary of OPIC as an example to demonstrate how to apply ontology terminologies to guide data capture.

In Chapter 5, We describe VISAGET’s design and implementation and how it is applied to automate matching between patients and clinical trials in Trial Prospector.

In Chapter 6, We present Active Dashboard for information summary, data navigation, and data query. Active Dashboard integrates all these tasks together concisely by defining states for patient data. Active Dashboard will be illustrated using results of DBSDB.

After that, we talk about Web Interface Driven Development (WIDD) in Chapter 7. We use programming interface of DBSDB as an example to present processes, roles and responsibilities, and iteration cycles of WIDD.

At last, we conclude this dissertation and discuss the possible improvements for future work in Chapter 8.
Chapter 2

Background

To better understand the work in this dissertation, in this chapter, we first systematically describe the background information for Trial Prospector, OPIC, and DB-SDB respectively. Then in later chapters, we will separately review related background information in the first section of each chapter.

2.1 Trial Prospector

Trial Prospector is developed for UH Seidman Cancer Center to address the challenges of finding eligible cancer patients for oncology clinical trials.

2.1.1 Clinical Trials

A clinical study involves research using human volunteers that is intended to add medical knowledge. There are two types of clinical studies: observational studies and clinical trials. In a clinical trial, participants receive specific interventions according to the research plan or protocol created by the investigators. These interventions may be medical products, such as drugs or devices, procedures, or changes of behaviors of participants. Some clinical trials compare a new medical
approach to 1) a standard one that is already available, 2) a placebo that contains no active ingredients, or 3) no intervention. Other clinical trials compare interventions that are already available to each other. When a new product or approach is being studied, it is not usually known whether it will be helpful, harmful, or no different than available alternatives (including no intervention). The investigators try to determine the safety and efficacy of the intervention by measuring certain outcomes in the participants. For example, investigators may give a drug or treatment to participants who have high blood pressure to see whether their blood pressure decreases. Large number of CTs are conducted at the international level, 153, 172 of which are registered in website clinicaltrials.gov with locations in 184 countries.

2.1.2 Eligibility Criteria

A CT is conducted according to a research plan as known as protocol. The protocol is usually designed by the principal investigator (PI) to answer specific research questions like the reason of conducting the CT, who can participate in the CT, the number of samples needed, length of the CT and so on. For us, the most important information that protocol provides is the standards outlining who can participate in the CT, called eligibility criteria. Eligibility criteria define a set of specific characteristics shared by all subjects of interest. These characteristics determine the rules to match patient to a CT. Some CTs seek participants who have the illness and conditions to be studied and others look for healthy people. And some CTs limit to a specific group of people who are asked by researchers to enroll. The factors that allow someone to participate in a CT may include demographic information such as gender, age, race, and living address, as well as lab test results such as primary diagnosis, white cell count, creatinine, and so on.
2.1.3 Challenges of Subject Recruitment in Clinical Trials

Clinical trials have to recruit enough participants to achieve its goals. However, a recent analysis [5] of NCI-sponsored clinical trials showed that many did not complete accrual, with 57% of the aborted studies failing to meet accrual goals. Researchers recently summarize three categories of barriers to the subject recruitment in clinical trials: system or organizational related barrier, individual barrier, and CT design related barrier [6]. In this thesis, our work focuses on the third barrier that is commonly shared by CTs. Protocol of CT is unstructured text, manually written by experienced protocol author or committee of clinical experts. Embedded in protocol, some eligibility criteria are complex and difficult to explain to patients, such as creatinine clearance requires a complex equation to calculate its value, especially in some cases, one eligibility criterion are dependent on other criteria. It takes oncologists a lot of time to compare patient information with eligibility criteria of clinical trials. Besides, there are too many trials for physician to remember. As far as we know, just for breast cancer, there are more than 20 CTs in Seidman cancer center. It is not practical for physicians to know all these CTs and remember their eligibility criteria and recommend for their patients.

2.1.4 Informatics-based Approaches

Informatics-based approaches that incorporate formal representation of eligibility criteria, data from EHR systems, and accurate trial matching algorithms are important for automating many phases of the trial recruitment workflow and efficiently matching trials with patients [7, 8]. In addition, default matching of all visiting patients to available trials by an automated matching tool can help improve the participant diversity across economic and social categories [9, 10]. Automated matching tools can also be integrated with existing data management systems used
in health care institutions (eg, patient scheduling systems) to leverage available resources while streamlining the trial recruitment process [11, 12]. An important challenge for computational representation of trial information is the lack of suitable data entry interface that not only facilitates structured data entry, but also reduces the burden on research staff through use of intuitive functional components. Trial Prospector addresses the challenge for computational representation of eligibility trials through the development of the “Trial Builder” visual interface to facilitate entry of trial information. We build a pipeline in Trial Prospector to assemble the tasks of selecting target trials, selecting target patients, matching patients to trials, and reviewing reports step by step to satisfy the clinical requirements to match specific subset of patients to specific subset of clinical trials.

2.2 OPIC: Ontology-Driven Patient Information Capturing System for Epilepsy

OPIC is created for UH Epilepsy Center for discharge summaries management.

2.2.1 Sudden Unexpected Death in Epilepsy (SUDEP)

For some people living with epilepsy, one important concern is the risk of SUDEP. SUDEP refers to deaths of epilepsy patients that are not caused by injury, drowning, or other known causes [13]. Studies [14] suggest that 1.16 cases of SUDEP happen for every 1,000 people with epilepsy in each year.

2.2.2 The Center for SUDEP Research (CSR)

CSR is a National Institute for Neurological Disorders and Stroke (NINDS) funded Center Without Walls for Collaborative Research in the Epilepsies. This milestone-
driven collaboration is composed of researchers from 14 institutions across the United States and Europe and brings together extensive and diverse expertise to tackle Sudden Unexpected Death in Epilepsy Patients (SUDEP).

2.2.3 Epilepsy Center of UHCMC

CSR requires patient data from all 14 institutions. As one of the participating and managing institution of CSR, Epilepsy Center of UHCMC is responsible for coordinating the patient data collection from all collaborating institutions. The document based way to data management is obviously not appropriate for CSR. A flexible and scalable patient data capture system (OPIC) is needed for all institutions to store patient data in a structured and consistent way.

OPIC is a web-based database application for patient information capture in Epilepsy Monitoring Units (EMUs). This system features a dynamic terminology library for epilepsy and seizure classification representing clinical knowledge of experts in the field. Using this library, our system can capture well-structured high quality patient data, compared to traditional paper-based or electronic form based approach in clinical practice. The enhancement of OPIC provides strong support for the secondary use of patient data for clinical research. Additionally, this ontology-driven method creates standard vocabulary that allows patient data to be managed and shared across multiple EMUs. The enhanced system manages patient data lifecycle such as patient creation, patient search, patient clinical data entry, attending signature, data export for printing, and data de-identification. The enhanced system is designed and implemented to be efficient, effective, maintainable, updatable, and user friendly, respecting the real-world workflow and user requirements in EMUs.
2.3 DBSDB: Database for Deep Brain Stimulation

DBSDB is designed and implemented for UH Deep Brain Stimulation program to manage DBS workflow and the data generated from the workflow.

2.3.1 Parkinson’s Disease

Parkinson’s disease (PD) is one of the most common neurodegenerative disorders. It is usually classified by clinical features such as bradykinesia, tremor and rigidity. According to Parkinson’s Disease Foundation, PD is currently affecting one million people in the United States and an estimated four million worldwide. It is estimated to grow substantially in the next 20 years according to a project conducted in the 5 most populous Western Europe nations and the world’s 10 most populous nations. The total number of individuals with PD is expected to be greater than 9 million in 2030 [53].

2.3.2 Deep Brain Stimulation

The Movement Disorders Center at the University Hospitals (UH) Neurological Institute in Cleveland offers Deep Brain stimulation (DBS) which is a treatment for Parkinson’s disease approved by Food and Drug Administration (FDA). In the past two decades, there are approximately 60,000 patients worldwide with PD have undergone this surgery, with an annual accrual of 8000 to 10,000 new patients [2]. Each patient who takes deep brain stimulation procedure generates huge volumes of data throughout the process. The accuracy, accessibility, legibility and processability of the large quantity of patient information and individual specific decision making directly determine the quality of patient care delivery and health outcomes.

The success of DBS therapy relies on the careful patient selection, precise surgery
plan, and optimal stimulation parameter settings [3]. A comprehensive system managing the entire lifecycle of DBS clinical data capture and decision support can be extremely complex. No off-the-shelf tools exist that could well document the electrophysiological data acquired for deep brain stimulation surgery, permanent electrode implantation location in the brain, or stimulation variables associated with therapeutic benefits. The previous electronic medical record (EMR) system adopted by Movement Disorders Center was for the use of general clinical management. It captures very limited data and lacks customized functionality, appropriate interface and data management flexibility to be part of the standard operating protocol for DBS.

We create a web-based, non-manufacturer dependent system, called DBSDB, for managing clinical data capture and management support for DBS. DBSDB is designed to integrate into the clinical workflow and mirror the actual process of DBS procedures. The system aims to capture longitudinal clinical data in three stages:

- Pre-operative registration and assessments,
- Intra-operative surgical data acquisition, and
- Post-operative evaluation and adjustment.

DBSDB centralizes the data collected from all stages of the procedure, organizes data by clinical visits, reduces data management complexity, and ultimately improves the quality and efficiency of the entire clinical workflow.
Chapter 3

Configurable Role-Based Access Control

Role-Based Access Control (RBAC) is a popular framework for implementing the security policy of an organizations enterprise information system. In RBAC, permissions are associated with roles, and roles are assigned to users. In most organizations, roles in enterprise systems often have their well-defined counterparts in the organizational structure, and are more stable than individuals who assume the roles. By decoupling access permissions from specific individuals through the roles intermediary, RBAC reduces management overhead and the potential for access errors in dynamic organizations [16]. In clinical departments, people act as many different roles such as attending, research fellow, clinical fellow, nurse, eeg technician and so on. People of different roles have different responsibilities and different level of accesses to the data. It is very natural to apply RBAC method to manage the patient data access.

In this chapter, we present a configurable role based access control method that can meet different levels or granularities of access control needs.
3.1 Background

3.1.1 Access Control

Access control is used to limit the actions or operations that can be performed by users in an information system [36]. Access control restricts what a user can do like what page they can see, what data they enter, and what data they can review and make changes. Nowadays, most information systems (IS) have access control module. With the application of access control, information systems try to prevent security issues that may be caused by activities or operations performed by users who are not supposed to. In addition to security issues, in CDMSs clinical management systems, uncontrolled user actions can also cause data quality issues. We have seen in some systems many users of different roles can access same set of forms. The later accessed users can erase the data entered by former users and cause data loss. It is easy to get confused about authentication and access control. It is important to make it clear that authentication is to verify and establish the identity of a system user and access control happens after user identity is successfully verified. In our systems, a user will have no access to our system if authentication fails at the first place. Access control policy and mechanism are two concepts are important to understand how to design and create access control module in information systems. Policies are guidelines that describes how accesses are controlled and how access decisions are made in access control systems at a high level. Mechanisms are low level system functions that are designed and created to implement a policy [36]. It is desirable in ISs to reuse access mechanisms implement different access policies.
3.1.2 Access Control Models

There are two classical access control models: mandatory access control or lattice based access control [37, 38] and discretionary access control [36, 39]. These two models have been created and recognized by researchers and practitioners for a long time. In MAC, access control is based on the user’s classifications or security levels. Each user is assigned with a security level and reading operations is only allowed from high level to low level and writing operations is only allowed from low level to high level. In DAC, access to the information is based on the user’s identity and authorization rules that specify what data object a user can access in what mode (own, write, or read). MAC is created from rigid environments like military and DAC is created from cooperative autonomous requirements for people like academic researchers. Neither of them can satisfy the needs of commercial environments [36].

3.1.3 Role Based Access Control

Following by the two classical models MAC and DAC introduced above, role based access control is introduced and researchers have claimed that its mechanisms are more general than other two classical models and can even be configured to simulate the functionalities of the traditional methods [40]. In role based access control, the ability to perform an action such as open a web page, create a new record, or edit an existing record, is defined as a privilege. A user is a human being or an autonomous agent. A role is a job function or job title within the organization with some associated semantics regarding the authority and responsibility. Multiple privileges can be assigned to a role and roles can be assigned to users. Therefore, a user with a role is allowed to perform actions that are defined in privileges assigned to the user’s role.
In clinical environments, a number of roles is naturally defined by their job titles such as: nurse, fellow, staff, attending, and so on. It is intuitive to choose RBAC to systemically manage users’ privileges and activities in clinical data management systems. We apply RBAC to almost every CDMS we develop. One fact we find after creating several CDMSs is different systems have different needs about access control. Although all systems capture and query clinical data, they have very different levels or granularities of needs about access control. We categorize these needs into three levels from high to low: role oriented, privilege oriented, and core specific privilege oriented RBAC. In role oriented RBAC, system operations is separated clearly into different components. There is no need to define privileges because role name is enough to decide what users can and need to do. Privilege oriented RBAC is the typical RBAC we talk about using privileges to restrict roles and using roles to restrict users. In other CDMSs, clinical data are maintained by different cores. In addition to typical RBAC, we need to introduce core model and only users in a specific core can access the data that is managed by the core. To address all these needs, we create a configurable role based access control software package. In the following section, we talk about the details about the design and implementation of cRBAC.

Figure 3.1 below shows the conceptual position of our role based access control module in the context of clinical data management system. User is authenticated first and then will try to perform some actions before they can do that we will send that action request to a RBAC module to authorize. If the authorization is approved, action continues. Otherwise, the action request is denied.
3.2 Design and Implementation

To begin data modeling, we first formally define all concepts we have in the context of CDMS:

1. User (U) – User is a human being that can perform various operations in our clinical data management systems;

2. Role (R) – Role is a title or job that reflects the ability and responsibility of a user. In clinical data management systems we have two categories of roles: functional roles and administrative roles. Administrative roles are responsible for the management of other roles;

3. Privilege (P) – Privilege is the basic unit that defines what operation is allowable or enabled in it;

4. Core (C) – Core is the natural separations of the clinical setting where our CDMS is deployed in. A core can be a department or a division, e.g. in DBS program, there are administrative core, imaging core, neuropsychology core, and so on;
5. Privilege Assignment (PA) – Privilege assignment happens between privilege and roles. Assigned privileges of a role allows the role to be able to perform all actions that are defined in each of the privileges;

6. Role Assignment (RA) – Role assignment happens within Role, Core, and User. A user can perform all actions that is allowed by any one of assigned roles.

Figure 3.2 demonstrates the relationships between all concepts above.

Database tables are created from the concepts and Table 3.1 displays the table names and main fields in each table. In privilege assignment table, PID is the foreign key of privileges table and RID is the foreign key of roles table. In role assignment table, RID is foreign key of roles table, CID is the foreign key of cores table, and UID is the foreign key of roles table. Role assignment does not require CID to be present.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Name, UID</td>
</tr>
<tr>
<td>Role</td>
<td>Name, RID</td>
</tr>
<tr>
<td>Privilege</td>
<td>Name, PID</td>
</tr>
<tr>
<td>Core</td>
<td>Name, CID</td>
</tr>
<tr>
<td>Privilege Assignment</td>
<td>PID, RID</td>
</tr>
<tr>
<td>Role Assignment</td>
<td>RID, CID, UID</td>
</tr>
</tbody>
</table>

*Table 3.1: Critical fields in concepts*
3.2.1 Privilege Assignment Table

Treating roles as objects and privileges as attributes, we create the following table to assign privileges to roles as in Figure 3.3. Each column is a privilege and each row represents a role. Checked privileges in each row are assigned the related role. For example, as shown in the picture, Role 1 has Privilege 1, 2, 3, and 8 while Role 7 has Privilege 3, 6, and 7.

![Roles and Privileges](image)

*Figure 3.3: Relation table that assigns privileges to roles*

3.2.2 Role Assignment Table

Similarly, we create the following table to assign roles to users depicted in Figure 3.4. Each column is a role and each row is a core. Semantically different from privilege assignment, checked roles in each row are not assigned to the related core. Instead, this table is placed in each user’s profile page. Therefore, the context here has the knowledge of a current user and checked roles in each row means that current user are assigned with those roles in the related core. For example, in Figure 3.4, current user has Role 1 and 2 in Core 1, Role 4 and 5 in core 2, Role 6 and 7 in core 3, and Role 8 in core 5. The user
has no role in Core 4.

<table>
<thead>
<tr>
<th>Roles Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role 1</td>
</tr>
<tr>
<td>Core 1</td>
</tr>
<tr>
<td>Core 2</td>
</tr>
<tr>
<td>Core 3</td>
</tr>
<tr>
<td>Core 4</td>
</tr>
<tr>
<td>Core 5</td>
</tr>
</tbody>
</table>

*Figure 3.4: Relation table to assign roles in cores to a user*

### 3.2.3 Three Access Control Mechanisms

With above design, we implement three access control mechanisms: authorize by role, authorize by privilege, and authorize by privilege in core.

- **Authorize by role** – In Trial Prospector, privilege assignments are completely exclusive within roles. No roles share the same privileges. In these systems, there is no need to define privileges. To authorize a user operation request, we just need to ask if current user has required role assignment. This mechanism can be expressed formally in ruby method for a user model as:

  ```ruby
  hasRole?(roleName).
  ```

  We also provide a simple algorithm for this mechanism in algorithm 1

- **Authorize by privilege** – In OPIC, privileges are shared within different roles. To effectively authorize user operations, we traverse all assigned roles to search for the required privilege. This mechanism can be expressed formally in ruby method as:
Algorithm 1: Pseudocode for access control mechanism 1: authorize by role.

\[\text{hasPrivilege?}(\text{roleName}).\]

Algorithm for privilege based authorization is formally described in algorithm 2

Algorithm 2: Pseudocode for access control mechanism 2: authorize by privilege.

- Authorize by privilege in core – In DBSDB, system are divided into multiple independent cores. To decide if a user has enough privilege to perform an action, we not only need check if the required privilege belongs to the user but also need to check if the privilege is assigned to the user in the required core. We can express this mechanism as follows:

\[\text{hasPrivilegeInCore}(\text{privilegeName}, \text{coreName}).\]
Algorithm 3 formally describes the details about how to authorize by privilege in core.

Algorithm 3: Pseudocode for access control mechanism 3: authorize by privilege in core.

3.3 Results

3.3.1 Role-based Access Control in Trial Prospector

Trial Prospector is a tool created for automating the matching between patients and cancer clinical trials. It provides a interactive interface to facilitate end users to build eligibility criteria of clinical trials. Trial Prospector also builds pipeline of matching including: clinical trials review, patients review, and generate matching reports. To facilitate the workflow, three roles are created: Physician, Trial Manager, and SystemAdministrator. Privileges and roles assignment interface are displayed in top half of Figure 3.5:
• Add User – This privilege allows user to perform operations like adding users, editing users, and removing users. This privilege also enables user to manage privilege assignments and role assignments;

• Manage Trial – With this privilege, a user can build create clinical trials and build eligibility criteria for clinical trials;

• Create Report – This privilege enables users to match patients data against clinical trials;

• View Report – With this privilege, user can review the generated matching reports between patients and clinical trials.

Role assignment interface is presented in the bottom half of Figure 3.5

• System Administrator has “Add User” privilege and can manage users and role based access control module;

• Clinical Trial Manager can create clinical trials and build eligibility criteria with privilege “Manage Trial”;

• Physician can trigger matching process between patients and clinical trials with privilege “Create Report” and “View Report”.

Because the privilege assignments are exclusion within all three roles in Trial Prospector, we use mechanism 1 – authorize by role to control users’ accesses.

3.3.2 Role-Based Access Control in Ontology-Driven Patient Information Capturing System for Epilepsy

OPIC is a tool created for generating discharge summary for epilepsy center. Eight (8) roles and 11 privileges are defined in OPIC.

Details about privileges assignments are displayed in top half of Figure 3.6:
Figure 3.5: Interface for privilege assignment and role assignment in Trial Prospector

- “Create Patient” allows user to create new patient record;
- “Create Report” allows user to create new patient report record;
- “View Report” allows user to view patient report;
- With “Edit Report” privilege, users can enter data to patient reports;
- “Sign Report” gives users the ability to put their signatures on the report;
- Privilege “Delete Report” allows users to delete patient reports;
- “Add User” privilege allows user to perform operations like adding users, editing users, and removing users. This privilege also enables user to manage privilege assignments and role assignments;
- “View Research Data” enables the action to view de-identified patient reports;
• “Manage Attending Data” allows users to print patient reports for clinical attendings.

Role assignments are presented in the bottom half of Figure 3.6

• Clinical Attending can view patient reports, edit patient reports, and sign reports.

• Clinical and research fellow can view patient reports and enter data to patient reports and clinical fellows can sign reports, too.

• EEG Tech can create new patients and new patient reports.

• Secretary can view patient reports and print reports for clinical attendings.

• System administrator can manage users.

• Other users in the system can view patient reports.

OPIC adopts mechanism 2 of access control – authorize by privilege, since its roles have overlapped privilege assignments and there is no core concepts in the system.

3.3.3 Role-based Access Control in Database for Deep Brain Stimulation

DBSDB is a clinical data management system implemented for Deep Brain Stimulation to manage its clinical workflow and all clinical data that is generated from the workflow. Deep brain stimulation are divided into six cores: administrative core, clinical core, imaging core, neuropsychological core, surgery core, and adverse events core. Five roles and 11 privileges are defined in DB-SDB. Privileges are as follows:
Figure 3.6: Interface for privilege assignment and role assignment in OPIC

- Global is a special privilege only designed for “authorize by privilege in core” mechanism. Role with privilege global is applicable to all cores;
- A role with view patient data privilege can access patient profile page;
- Enter patient data allows users to edit patient page to do data entry;
- Review patient data is higher privilege than “Enter patient data” that allows user to review other users’ data entry and make edits if needed;
- “View Full Data” means user can view all data about patients including identified data;
- On the other hand, ”View De-identified Data” means user can only view data without PHI;
- ”View Number” is lower privilege than ”View De-identified Data” that only allows users to view the total number of patients without seeing any details of patient data;
• “Add User” is the same administrative privilege as mentioned above in Trial Prospector and OPIC;

• “Create Project for User” is another administrative privilege for research project creation in DBS;

• “Manage Project” allows user to perform project related operations like add a member and edit the project content;

• “View Project” is usually the privilege for a project member allowing them to view information from their projects.

DBSDB uses the third mechanism we create: authorize by privilege in core. As depicted in Figure 3.7, a user can be assigned with 5 roles in 6 cores which is 30 combinations in total. We describe the roles in details:

• Clinical attending can view, enter, and review patient data.

• Clinical staff has the same privileges as clinical attending except they cannot review patient data which prevents them from editing other clinical staff’s data entries.

• Project Managers can view full data in their project and manage their own projects.

• Project member can only view number of patients in their projects and view project information.

• System administrator in this DBSDB has all privileges.

3.4 Discussion

Access control is an important module of information systems that ensures users to access functionalities properly to prevent security issues and im-
prove performance. Following two classic access control methods mandatory access control and discretionary access control, role based access control is introduced by researchers and become more and more popular [36]. Researchers proved that RBAC is more general than traditional access control models and are appropriate for commercial organizations [40]. In this chapter, we introduce the design for a generic type of role based access control called configurable Role Based Access Control. We implemented 3 access control mechanisms: authorized by role, authorized by privilege, and authorized by privilege in core in cRBAC. We apply cRBAC and these three mechanisms in our three clinical data management systems: Trial Prospector, OPIC, and
DBSDB. Trial Prospector manages 4 privileges, 3 roles and 68 users. 85 active clinical trials, 1011 eligibility criteria, and more than 5,000 matching reports between patients and clinical trials are created and managed in Trial Prospector; OPIC has 100 active users with 8 roles and 9 privileges and more than 1,100 discharge summaries are created in OPIC; DBSDB has 16 users, working in 6 cores, 5 roles, and 11 privileges managing more than 1,000 data forms of 264 patients.
Chapter 4

Ontology Guided Data Capture

Data capture is one of the most important tasks for clinical data management systems because it directly affects data quality. In many of today’s clinical settings such as epilepsy center, deep brain stimulation program, and cancer center, paper-based or electronic form-based data capture methods are still in use. This way of data capture is prone to manual error and may produce unstructured data of low quality with no assurance of accuracy and consistency. Data quality issue has been acting as a critical barrier for secondary use health data. Poor data quality also tremendously increases the cost of later data curation and management.

This challenge can be addressed by an ontology guided web-accessible and flexible data capture system that supports data re-usability, sharing, and integration. Ontology is used to represent conceptual structure of of a specific domain by specify the entities (or concepts), their attributes and relationships among the entities. Nowadays biomedical ontologies are have been increasingly enabled in biomedical applications for query of heterogeneous biomedical data, data integration and exchange, natural language processing (NLP), decision support and reasoning [17, 18]. In this dissertation, the major role of
biomedical ontologies is to serve as the knowledge sources supporting data capture to provide standardization and reusability.

In this chapter, we present OPIC (version 2.0) – a web-based database application for ontology-guided patient data capture in Epilepsy Monitoring Units (EMUs).

4.1 Background

4.1.1 The Center for SUDEP Research (CSR)

OPIC is created for the Center for SUDEP Research as a flexible and scalable informatics infrastructure for interoperable, consistent, and error-free patient data collection. SUDEP stands for Sudden Unexpected Death in Epilepsy Patients which accounts for approximately 1 in 5 deaths in patients with epilepsy, but its cause remains unexplained. The rate of sudden unexpected death in people with epilepsy is as much as 24 times higher than the rate of sudden death in the general population, with an estimated 2,750 cases of SUDEP in the U.S. alone every year [15]. CSR is a National Institute for Neurological Disorders and Stroke (NINDS) funded Center Without Walls for Collaborative Research in the Epilepsies. This milestone-driven collaboration is composed of researchers from 14 institutions across the United States and Europe and brings together extensive and diverse expertise to tackle SUDEP. In such a large scale, multi-center study, CSR requires the ability to consistently capture data from all 14 participating institutions and store the data in a common format allowing real-time access for clinical investigators.
4.1.2 Role of OPIC

As one of the informatics tools of CSR, OPIC is responsible for the collection of patient phenotypic data including demographic, chief complaint, classification of paroxysmal episodes, history, past and current medications, positive findings on physical and neurological examination, evaluation, impression and plan, seizures or events, and conclusion and recommendations.

OPIC is planned to be deployed at each institution of CSR and data collection is done locally at each site. Collected data is then de-identified and pushed over to the center data repository of CSR. With the help of a cross cohort query interface for SUDEP called MEDCIS, clinical researchers can get access to the data pushed over by all sites. Researchers can perform patient cohort identification and download the data in de-identified pdf format.

4.2 Method

4.2.1 Use Cases

As we have discussed in Chapter 3 Configurable Role based Access Control, OPIC has the following 8 roles: Clinical Attending, Clinical Fellow, EEG Reader, EEG Tech, Other, Research Fellow, Secretary, and System Administrator. Figure 4.1 shows the most important use cases of OPIC.

(a) Create Report: Actor EEG Tech is responsible for the creation of patient and patient report in OPIC. In epilepsy center of university hospitals, EEG Technician has access to patient demographic records. Therefore, they can ensure the accuracy of the data needed at the moment of patient creation. Besides, at each patient visit, a patient report is created to
Figure 4.1: Use cases in OPIC

record all the information related to that visit. EEG Tech creates the report with necessary information like admission date and discharge date. OPIC also provides a function for them to copy from existing patient reports to save efforts. They also need the privilege to view the reports they have created to verify the results of their work.

(b) View Report: View report is most basic functionality in OPIC. All actors except secretary in Figure 4.1 can view report with different purposes. EEG Tech needs to view report verify they create the report with correct initial information; Research clinical fellow need to view patient reports to verify the data they enter; and Clinical Attending reviews reports before they finalize and sign the reports.
(c) Print Report: In OPIC, report can be exported and printed in PDF format. Report printing serves multiple purposes. Clinical attending, clinical fellow, research fellow, and secretary can print reports.

(d) Edit Report: Edit report is the activities to enter patient data into OPIC system. Data entry of OPIC is mainly the responsibility of clinical and research fellows. Clinical attendings also have the ability to edit a report because they need to review and confirm all the data entries from fellows. They can update the reports if they find some problems about the entries made by fellows.

(e) Sign Report: Only clinical fellow and attending can sign reports. By signing a report, clinical fellows claim that to the best of their knowledge all the data entered is accurate. By signing a report, clinical attendings claim that to the best of their knowledge, they have reviewed and find the data entered is accurate.

(f) De-identify Report: In compliance with HIPAA (the federal Health Insurance Portability and Accountability Act of 1996), patient data must be de-identified before it can be used for research purpose and be accessed by other investigators. Clinical attending or fellow can de-identify patient reports.

4.2.2 OPIC Report Structure

OPIC manages patient data lifecycle from patient creation, patient search, patient clinical data entry to attending signature, data export for printing, and data de-identification and so on. It is designed and implemented to be efficient, effective, maintainable, updatable, and user friendly, respecting the real-world workflow and user requirements in EMUs. Figure 4.2 show one
report for a fictional patient Skotnica Aaron to demonstrate the structure of OPIC reports.

Figure 4.2: OPIC report structure: 12 independent data sections and one operation panel

- Demographic section records patient’s demographic information together with admission and discharge date, attending and fellow information, referring physician and institution.

- Chief complaint is a concise statement describing the symptom, problem, condition, diagnosis or other factor that is the reason for the patient visit.
• Classification of paroxysmal episodes formally records the details of the diagnosis.

• History section records all related history information including seizure onset and evolution, possible risk factors, family history, medical and surgical history, allergies and so on.

• Past and current medications section stores details about the medication patients taken both in the past and currently. Recording information includes drug name, dosage, daily dosage, start data, stop date and stop reason if the medication is taken in the past.

• Positive findings of physical and neurological examination section keeps records of the positive findings that can be exported as progress notes and entered into EMR systems. One field that is required in physical examination form is BMI which is also a field in SUDEP checklist.

• Evaluation section is divided into previous evaluation and current evaluation to record the evaluation results from various images or videos or EEGs.

• Impression and plan section is daily updated and can be exported as progress notes together with positive findings of physical and neurological examinations.

• TABLE1(Medication) is a table of medication information recorded in EMU every day.

• TABLE2(Seizure/Event) is a table of seizure or event information recorded during patient stay in EMU.

• Conclusion and recommendations section concludes the diagnosis and treatment to the patient visit.
• Signature section allows clinical fellows and clinical attendings to electronically put their signature on the report.

• Operation panel contains four buttons: progress note, discharge summary, de-identified summary, and epilepsy surgery form. Progress note button exports patient information, positive findings of physical and neurological examinations, and impression and plans to plain text. Discharge summary button export patient report to PDF format for printing or downloading. De-identify summary button leads user to the workflow to identify patient report and finally generate a de-identified patient report in PDF format. Epilepsy surgery form creates a form to capture surgery information that is related to the specific patient visit to EMU.

4.2.3 Ontology Guided Data Capture in Classification of Paroxysmal Episodes

Classification of Paroxysmal Episodes is the most important section of all 12 data sections in OPIC described above. It records the information that SUDEP researchers care most such as diagnosis information, epileptogenic zones, seizure semiology’s, lateralizing signs, etiology, and related medical conditions. The implementation of this section is also the most challenging because all these information need to be captured in a structured way that we can use these information to identify patient cohorts. To achieve this goal, we use the latest terminologies developed by Dr. Hans Lüders, the author of ”Lüders Classification” [42], to guide the implementation of this section. The terminologies include 51 epileptic seizures, 51 non-epileptic episode types, 72 epileptogenic zones, 75 etiologies, and 11 lateralizing signs. Details of the
terminologies are attached as appendix 1. We describe how to apply these terminologies in the following step by step.

**Step 1. Modeling of Classification**

![Diagram showing the modeling of epileptic seizure terminologies in episodes classification section.]

Figure 4.3: Modeling of epileptic seizure terminologies in episodes classification section.

We build data models with Episodes Classification in the center and ontology
models around it. Each ontology model has the same data schema: name, parent name, parent id, and has modifier. Intuitively, name is the concept name of the terminology and parent name is the name of the parent concept; Parent id is the id in the database table of the parent concept; Modifier defines additional attributes of a epileptic terminology. “has modifier” is a flag that indicates a terminology has additional attributes. Each of the ontology models: epileptogenic zone, seizure semiology, episode semiology, materializing sign, and etiology has a relation model that bridges it to the central classification model.

Possible modifier values from 1 to 9 represent 9 different categories of additional attributes for ontology terms in OPIC. Tables 4.4 shows the details for all these 9 categories of modifiers. Each modifier category has first level modifiers. Some categories such as category 1, 4, 5, and 9 have second level modifiers. Only those first level modifiers ending with star symbol have second level modifiers available for selection. Figure 4.4 shows the first modifier category working in episodes classification.

### Step 2. Import Ontology Terms

Importing ontology terms is a two-run process.

(a) First run is to create records for all terminology. The information captured at this time includes: name, parent name, and has modifier or not.
Parent id remains empty.

(b) Second run is to create the hierarchy of the terminologies: for each term, we find its parent term by its parent name and update its parent id with the id of the parent term.

Step 3. Build Widgets for Ontology Navigation and Selection

We create a widget called Tider, standing for terminology finder in OPIC for efficient term navigation and selection. Tider is a hierarchical, multi-level menu style widget recursively built from the terminologies in “Subclass Of” or “Is-A” relations. It supports composition, revision, and update of epilepsy disease classification. The following algorithm 4 demonstrates how we create Tider for ontology models of OPIC.
**Input:** Ontology model name such as EpileptogenicZone, Etiology, and SeizureSemiology

**Output:** Tider Widget

1. Create an empty un-ordered list Tider;
2. Get model handle M from model name;
3. Find all top-level concepts C of the M;
4. **for each concept in C do**
   5. Call method getDropdown(concept) to return a list consists of all descendant concepts of current concept;
   6. Add the returned list to Tider;
5. **end**
6. Return Tider;
7. getDropdown is defined as follows;
   - **Input:** Concept
   - **Output:** List, the element of this list can be a concept or a list of concepts
8. Initialize a empty list L;
9. **if Concept has no children then**
   10. add Concept to L;
   11. return L;
10. **else**
   11. add Concept to L;
   12. **for each children Child of Concept do**
   13. add list returned by getDropdown(Child) to L;
   14. **end**
15. return L;
16. **end**

**Algorithm 4:** Pseudocode for Tider creation for ontology models in OPIC.

Figure 4.5 shows the collection of OPIC Tiders for Epileptogenic Zone, Etiology, Lateralizing Sign, and Seizure Semiology in working environment. All items in a Tider can be clicked to select no matter they are at the intermediate level or at leaf level. User can follow the little arrow to the right of terms to navigate to lower level terms.
Figure 4.5: Collection of 4 Tiders for OPIC Terminologies
Step 4. Create Interactive User Interface for Data Entry

OPIC creates a interactive interface to facilitate the data entry of episodes classification. Click on "+ New" button (top part in Figure 4.6) opens the classification form as shown in the bottom of Figure 4.6. With the help of Tiders, use can quickly specify other fields. Figure 4.7 is an example of such a classification of paroxysmal episodes. We avoid to use free text to capture data as much as possible, but we still keep the flexibility in places like frequency description or related medical conditions. Although we have provide a data structure fore seizure frequency, there are still expressions (e.g., 3 times in the last 10 years) where our data structure cannot fit.

![Figure 4.6: New form for Episodes Classification](image)
4.3 Results

As one of the main informatics tool of CSR, OPIC has been deployed in 7 of the CSR participating institutions: UH case medical center, NYU, University of Iowa, Thomas Jefferson University, University College London, UCLA, and Northwestern University. In less than one year, OPIC has collected phenotypic data from in total 388 individual patient visits in all these sites of CSR. The ontology guided way of data capture enables MEDCIS – another tool of CSR [43] to build cross cohort queries using not only patient demographic data like age and gender but also clinical characteristics like semiologies and epileptogenic zones. Other functionalities implemented in OPIC but are not described in detail here include its integration with institution local authentication system, role based access control (mentioned in Chapter 3), electronic signature, de-identification workflow, and dashboard for site statistics.
4.4 Conclusion and Discussion

In this chapter, we present a ontology guided patient data capture system OPIC.

OPIC features a dynamic terminology library for epilepsy and seizure classification representing clinical knowledge of experts in the field. Using this library, OPIC can capture well-structured high quality patient data, compared to traditional paper-based or electronic form based approach in clinical practice. Besides, we create Tider to support composition, revision, and update of epilepsy disease classification. It tremendously reduces manual error like typos and inconsistencies.

With ontology providing data standardization, data capture in OPIC is accurate, consistent, and efficient that provides strong supports for the secondary use of patient data and empowers large scale, multi-institutional studies like CSR.
Chapter 5

VISAGET – Visual Aggregator and Explorer for Clinical Trials

Screening patients to find eligible subjects for clinical trials is a typical patient cohort identification problem. Currently this screening work is manually done by physicians. As we have discussed in Chapter 2. This matching process between patient data to clinical trials’ eligibility criteria is labor-intensive and has become a major barrier to the subject recruitment of clinical trials.

To address this issue, we create Trial Prospector, a point-of-care clinical trial matching system to automate the matching process between patients and clinical trials. Trial Prospector aims to improve the efficiency of subject recruitment for clinical trials.

In this chapter, we present the patient cohort identification method used in Trial Prospector: VISAGET – Visual Aggregator and Explorer for Clinical Trials.
5.1 Background

Clinical trials represent the evidence base for development of new cancer treatments. American Cancer Society and National Comprehensive Cancer Network endorse clinical trials as a component of high quality care and encourage patients to participate. Unfortunately, patient participation rate is low. Less than 5% of all cancer patients take part in clinical research studies [49, 50]. A key barrier to patient recruitment for clinical trials is the time and effort it takes for a physician to identify an appropriate clinical trial for a patient during routine clinical practice [51, 52]. The screening process usually includes: a) thinking of clinical trials in general; b) determining whether there is an available clinical trial for the patient’s disease and stage; c) reviewing specific clinical trial eligibility criteria to determine if the patient is eligible. VISAGET is created as a solution to this labor intensive process. VISAGET uses the following two resources.

5.1.1 Common Data Elements (CDEs)

CDEs are standardized terms for the collection and exchange of data. The National Cancer Institute (NCI) has developed its own CDE initiative to address the need for consistent cancer research terminology. The CDE Browser [44] provided by NCI allows public access for searching, viewing and downloading CDEs in Excel or XML format. To build eligibility criteria library for VISAGET, we extract CDEs about patient demographics, social status, behavior history, and laboratory tests.
5.1.2 VISual AGgregator and Explorer (VISAGE)

VISAGET’s interface design is adapted from VISAGE [45]. VISAGE is developed by our group in 2010 as part of the Multi-Modality Multi-Resource Environment for Physiological and Clinical Research (Physio-MIMI). VISAGE creates an ontology-driven framework for a federated approach to data integration. VISAGE’s interface design features include auto-generated slider bar, selection boxes, and built-in charting. VISAGE also builds administrative and query lifecycle management functionalities, such as role-based access control, auditing, query builder, query manager, and query explorer. We adapts and extends VISAGE with trial specific primitives, such as conditional branching and comparing expression to provide an interactive interface to model eligibility criteria for clinical trials.

5.2 Methods

In this section, we propose the modeling of core concepts in VISAGET: eligibility criteria ontology, trials, eligibility criteria, patients, patient data item, and matching. Eligibility criteria are embedded in protocol of a CT in the form of unstructured free text, which is not readable by computers. How to represent eligibility criteria in a proper way that computers can process them efficiently is a big challenge while more E-Screening tools are built to provide decision support for clinical research screening. A formal representation of eligibility criteria consists of 3 critical components: expression language, codification of eligibility concepts, and patient data modeling [46]. In VISAGET, we build eligibility criteria ontology for eligibility concepts codification reusing terms from existing terminological system; construct eligibil-
ity criteria expressions using interactive web-based user interface and store them in relational database; and model patient data as patient-eligibility concept associations. Hence, we can run eligibility criteria as SQL queries against patient data to automate the screening process of clinical trial recruitment.

5.3 Eligibility Criteria Ontology

To formally capture the common fields of an eligibility criterion and allow our screening system to reuse existing terminology system, we build an Eligibility Criteria Ontology (ECO) for criteria library construction.

5.3.1 Attributes of ECO

(a) Name — Full name of the criterion, e.g. White Blood Cell Count

(b) Short Name — Abbreviation of the full name for a criterion, e.g., WBC is the short name for White Blood Cell Count

(c) Category — Two values to categorize all criterion for current version of VISAGET: demographic and lab. Demographic criteria include gender, age, metastasis, TNM (Tumor, lymph Node, Metastasis) stage, TNM stage group, and so on. These criteria can be out of date when the screening is executing so that physicians can update these fields to update the matching report when they review the screening report of a patient. Other criteria are categorized as lab results criteria, which are done recently by the patients collected from lab tests system from hospitals or related organizations.

(d) Maximum and Minimum — Maximum and Minimum defines the possible value limit for a criterion if this criterion’s value type is continuous
like White Blood Cell Count.

(e) Unit — Unit is followed by the value of a criterion to make it meaningful.

(f) Value or Values — Value or values are possible options for a categorical criterion, such as gender’s values are male or female, and TNM Stage’s values could be T1N1M1, T1N2M1, and so on.

(g) Value Type — Four options exist for value types. They are continuous, boolean, categorical, and dynamic categorical. Simple examples for these value types are respectively age, gender, TNM stage, and diagnosis.

(h) Description — Description defines how we talk about this criterion in a natural language. This attribute for a criterion is critically important as we discussed above that an eligibility criterion could be very complex and hard to explain to patient or even to physician not directly in the field. Description in a way makes criterion easy to understand and meaningful to people without special training, e.g. description of WBC is “White blood cell count of eligible should be ?”, in which “?” is a place holder and will be replaced with the real values when rendered and the rendering is dependent on value type and value itself. We will talk about the details in next section.

(i) Step Size — This is a setting for continuous eligibility criteria. The range of a continuous eligibility criterion could be from 0 to 10,0000 like WBC, and it could also be from 1 to 120 like age. It is necessary to define a proper grain level to make it convenient and operational when end user who is often a physician or CT PI specifies the criteria limit values through a slider bar.
Figure 5.1 demonstrates an example of how ECO represents a typical eligibility criterion subject “White blood cell count,” which has a continuous value type with minimum 0 and maximum 10,000 of unit per micro liter. We keep the schema of ECO as simple as possible to maximize the reuse of existing terminological system such as NCI Common Data Elements (CDEs), SNOMED CT [47], and NCI Thesaurus [48] with minimal effort. Initial version of ECO consists of 32 CDEs extracted from NCI CDE Browser. Without losing flexibility, VISAGET also provides a function that allows manual entry for eligibility criteria terms, in addition to above systematic ontology-driven import.

5.3.2 Build Eligibility Criteria from ECO

A typical simple eligibility criterion (e.g. Age > 18 years old) is a statement that has noun phrase as subject and a simple fact or mathematical expression about the subject [46]. There are also some more complex eligibility criteria (e.g. Aspartate Aminotransferase (AST/SGOT) < 5 times of upper limit of normal if diagnosis is hepatocellular carcinoma (HCC)) that are constructed
from simple ones connected by logical connectors such as “if-then” and “if-then-else.” Once we have ECO ready, we can build real CT eligibility criteria from it. As discussed above, we have two types of eligibility criteria to build. We call them single criterion and group criterion in VISAGET.

(a) Single criterion involves only one ECO term. The building process involves two steps and the specific operations depend on value type of the term. Table 5.3 shows the specific operations needed to build single criterion.

<table>
<thead>
<tr>
<th>Value Type</th>
<th>Step 1</th>
<th>Step 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous</td>
<td>Select ECO term</td>
<td>Specify the lower limit and higher limit of the term</td>
</tr>
<tr>
<td>Boolean</td>
<td>Select ECO term</td>
<td>Choose one option from the two possible options</td>
</tr>
<tr>
<td>Categorical</td>
<td>Select ECO term</td>
<td>Check all eligible options for the term</td>
</tr>
<tr>
<td>Dynamic Categorical</td>
<td>Select ECO term</td>
<td>Enter required values for the term</td>
</tr>
</tbody>
</table>

(b) Compared to single criterion, group criterion involves more than one ECO terms and the number of terms depends on the relationships between all the terms. Current version of VISAGET support two types of relationship: IF and IF-ELSE. IF relationship involves two single eligibility criterion: IF A THEN B and IF-ELSE relationship involves 3 single eligibility criteria: IF A THEN B ELSE C. Table 5.2 shows the building process for this two kinds of group eligibility criteria. In later section relationship type and group type are exchangeable referring to IF or IF-ELSE.
Table 5.2: Build group eligibility criterion from eligibility criteria ontology

<table>
<thead>
<tr>
<th>Group Type</th>
<th>Template</th>
<th>Step 1</th>
<th>Step2</th>
<th>Step3</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF</td>
<td>IF A THEN B</td>
<td>Choose A, B from ECO</td>
<td>Build A, B as single criterion</td>
<td>Group A, B</td>
</tr>
<tr>
<td>IF-ELSE</td>
<td>IF A THEN B ELSE C</td>
<td>Choose A, B, C from ECO</td>
<td>Build A, B, C as single criterion</td>
<td>Group A, B, C</td>
</tr>
</tbody>
</table>

5.3.3 Modeling of Clinical Trials

As described above, we know that a CT protocol answers why to conduct such CT, who to participate in this CT, how to protect participants and so on. However, only eligibility criteria and some other essential information such as Trial Number, PI, Research Nurse, and Phase make sense to us. In VISAGET a trial is a record of essential information with many single and group eligibility criteria associated with it as Figure 5.2 demonstrates.

(a) Trial Number — is the unique key of a CT and differentiates it from other trials.

(b) PI — is the principal investigator of the CT and the main contact person.

(c) Research Nurse — are response for CT coordination and data management, with a central focus on managing subject recruitment and enrollment.

(d) Phase — CTs are conducted in a series of steps called phases. There are I, II, III, and IV four phases and each of them is designed to answer a separate question.
The building of a trial in VISAGET is basically to keep building eligibility criteria and add them to the trial. Eligibility criteria of a CT are logically in “AND” relation, which means one person is allowed to participate in a CT only if he or she satisfy all eligibility criteria of the CT.

5.3.4 Patients in VISAGET

The collecting of patient data — both demographic and lab tests are out of the scope of this thesis. We assume that we already all patient data exist in our relational database and we know the mappings between patient table column and ECO terms. In the deployed Trial Prospector, patients are identified by name and medical record number. In this thesis we use pseudo id to identify patient. Patient data are mapped to existing ECO, which means, for each ECO term in the library, we search patient information to see if related information exists based on the mappings information. If yes, an association between patient and that term and the value of patient information is added to this patient, called patient data item. Figure 5.3 demonstrates the patient model in VISAGET, in which one patient consists of a set of patient data items and each patient data item has a related ECO term.

*Figure 5.3: Patient model in VISAGET*
5.3.5 Matching Patients with Clinical Trials

With the modeling CTs’ eligibility criteria and patient data, the matching process is straightforward. As demonstrated in Figure 5.4, Matching between Patient and CT is transformed to be matching between patient data item and eligibility criterion identified by the same ECO term.

![Diagram of matching process](image)

Figure 5.4: Match patient data to a clinical trial

A patient is eligible for a CT only if all patient data items match with the specific eligibility criteria of the CT. One possible scenario is that patient information is not always complete. In that case some eligibility criteria of a CT cannot find the related patient data items. In that happens we treat this patient also eligible for that CT to avoid false negative, because E-Screening result is not the final result, manual review will follow this E-Screening to finalize the participants.
5.4 Implementation of VISAGET

In this section, we focus on the details of VISAGET implementation, including the development environment we use, building of ECO, interactive CT builder, matching algorithm between patients and clinical trials, and screening report review.

5.4.1 Build Eligibility Criteria Ontology

There are two ways to build eligibility criteria ontology: import from existing file in xml format or create from the web interface.

Import ECO from existing file

The input files need to be in xml format. The content of xml file consists of series of the following tag Figure 5.5:

```
<Term>
  <name> </name>
  <category> </category>
  <maximum> </maximum>
  <minimum> </minimum>
  <value> </value>
  <unit> </unit>
  <description> </description>
</Term>
```

*Figure 5.5: Node syntax in XML file for ECO*

From the figure above, we can see all necessary fields of ECO. We can extract all these ECO term information with an XML parser.
5.4.2 Build ECO from Common Data Elements

We extract the following 32 common data elements of 4 areas: demographics, social status, behavior history, and laboratory tests:

(a) Demographic — Date of birth, Gender, Ethnicity, Race, Maternal ethnicity, Maternal race, Paternal ethnicity, Paternal race

(b) Social Status — Education level, Marital/Partner Status

(c) Behavior History — Current tobacco use, Past tobacco use, Age started tobacco use, Age stopped tobacco use, Type of tobacco use, Current drinker, Past drinker, Age started drinking, Age quit drinking, Current drug user, Drug type used

(d) Laboratory Test — Creatinine, Aspartate Aminotransferase (ASAT/SGOT), Alanine Aminotransferase (ALAT/SGPT), Total Bilirubin, Sodium, Potassium, Hemoglobin, Hematocrit, White blood cell count (WBC), Neutrophils, Lymphocytes, Eosinophils, Platelet And 2 more terms for demographics, 5 more terms for health status, and 6 more terms for Laboratory tests are manually added:

(e) Demographic — Age, Address

(f) Health Status — Life Expectancy, Metastasis, TNM Stage, Stage Group

(g) Laboratory — International Normalized Ratio (INR), Prothrombin Time (PT), Activated Partial Thromboplastin Time (aPTT), Creatinine Clearance, Absolute neutrophil count (ANC), Alkaline

5.4.3 Coverage of Semantic Classes for Eligibility Criteria

According to one recent study about Semantic Classes for Eligibility Criteria [34], there are 27 semantic classes of clinical research eligibility criteria,
forming 6 exclusive topic groups, which are Demographics, Health Status, Treatment or Health Care, Diagnostic or Lab Tests, Ethical Consideration, and Lifestyle Choice. ECO terms distribute in four groups of demographics, Health Status, Diagnostic or lab tests, and life style choice. 14 of total 16 semantic classes of these four groups are covered by ECO terms (term in the bracket in the following list):

(a) Demographics — Address (Address), Age (Age), Gender (Gender), Literacy (Education level), Ethnicity (Ethnicity, Maternal ethnicity, Paternal ethnicity), Special Patient Characteristics (Marital/Partner Status)

(b) Health Status — Life Expectancy (Life Expectancy), Neoplastic Status (Metastasis), Disease Stage (TNM Stage, Stage Group), Disease, Symptom or Sign (Primary Diagnosis)

(c) Diagnostic or lab results (White blood Cell Count, ANC, INR, etc)

(d) Life Style Choice — Addictive Behavior (Current tobacco use, Current drinker, Past drinker, Current drug user, etc)

After inspecting the remaining 13 semantic classes of eligibility criteria, VISAGET can support them immediately after adding them into the system from criteria creation form. For example, consent can be modeled as a boolean criterion, which has two possible values yes and no. With this coverage, we demonstrate that VISAGET is capable or at least has the potential to be able to represent eligibility criteria of all semantic classes. We retrieve 15 active trials we from a clinical trial management system named as On-line Clinical Oncology Research Environment (OnCore) [19]. All of them are built with our interactive trial builder. There are in total 148 eligibility criteria involved in these 15 trials. 10.1% of them are about patient demographics and 89.9% are about lab tests. There are 94.6% simple criteria and 5.4% complex
criteria composed with logical expressions. In table below, Table 3 demonstrates the building details for trial CALGB80702, which is a phase III trial Of 6 Versus 12 treatments of adjuvant eolfox plus celecoxib Or placebo for patients with resected stage III Colon Cancer. There are 8 eligibility criteria about this trial and Creatinine Clearance is a complex criterion.

<table>
<thead>
<tr>
<th>Protocol Description</th>
<th>ECO Term</th>
<th>Widget Visualization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age $\geq$ 18 years old</td>
<td>Age</td>
<td><img src="image" alt="Age" /></td>
</tr>
<tr>
<td>The cancer should not be metastasis</td>
<td>Metastasis</td>
<td><img src="image" alt="Metastasis" /></td>
</tr>
<tr>
<td>Primary Diagnosis is Colon Cancer</td>
<td>Primary Diagnosis</td>
<td><img src="image" alt="Primary Diagnosis" /></td>
</tr>
<tr>
<td>Absolute neutrophil count (ANC) $\geq$ 1500/ul</td>
<td>ANC</td>
<td><img src="image" alt="ANC" /></td>
</tr>
<tr>
<td>Platelets count $\geq$ 100000/ul</td>
<td>Platelets</td>
<td><img src="image" alt="Platelets" /></td>
</tr>
<tr>
<td>Total Bilirubin $\leq$ 1.5 \times upper limit of normal range</td>
<td>Total Bilirubin</td>
<td><img src="image" alt="Total Bilirubin" /></td>
</tr>
<tr>
<td>Creatinine $\leq$ 1.5 \times upper limit of normal range</td>
<td>Creatinine</td>
<td><img src="image" alt="Creatinine" /></td>
</tr>
<tr>
<td>If Creatinine is out of range (greater than upper limit of normal range), Creatinine Clearance $&gt; 60$</td>
<td>Creatinine Clearance</td>
<td><img src="image" alt="Creatinine Clearance" /></td>
</tr>
</tbody>
</table>

**Figure 5.6: Mappings between protocol description, ECO term, and visualization widget**

### 5.4.4 Create or Edit from System Interface

VISAGET also provides interface to create ECO terms and to update the existing terms Figure 5.7.

Each attribute entry consists of label and input widget. The label gives instructions about the creation of an eligibility criterion term.
5.4.5 Interactive Interface for Building Eligibility Criteria

VISAGET provides a flexible interactive interface called trial builder shown in Figure 5.8 that supports eligibility criteria building and managing. Trial builder consists of 5 components: eligibility criteria browser, panel for eligibility criteria building, panel for eligibility criteria description, group function buttons, and buttons to save eligibility criteria.

(a) Eligibility Criteria Browser — To add a single eligibility criterion to a specific clinical trial, first of all we need locate the definition of that specific criterion. To accomplish this task, we design a keyword-based criteria browser that queries the input keyword against the names of all criteria in the systems. Obscure query is supported, for example, enter keyword “ag,” relevant criteria “Age,” “Gender,” “TNM Stage,” and “Stage Group” is listed as available candidates.

(b) Eligibility Criteria Building Panel — Once the target criterion is found from eligibility criteria panel, with one click on the “plus” icon in front of display of that criterion, that criterion is added to the current trial
Figure 5.8: Interactive trial builder

and a widget representing this criterion will automatically added criteria building panel. Different criteria have different visualization widgets based on their value type defined in ECO. Table shows all mappings between value types and widgets. After adding all criteria, specifying the desired values, and clicking the save trial button, one trial with well-defined computable eligibility criteria is successfully built.

<table>
<thead>
<tr>
<th>Value Type</th>
<th>Visualization Widget</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous</td>
<td>Slider Bar</td>
<td>Specify the lower limit and higher limit of the term</td>
</tr>
<tr>
<td>Boolean</td>
<td>Radio Button Group</td>
<td>Choose one option from the two possible options</td>
</tr>
<tr>
<td>Categorical</td>
<td>Checkboxes</td>
<td>Check all eligible options</td>
</tr>
<tr>
<td>Dynamic Categorical</td>
<td>Text to add dynamically</td>
<td>Enter required values for the term</td>
</tr>
</tbody>
</table>

(c) Group Button — When we review the protocols, we find out some criteria, which is different than “Age is older than 18 years.” However,
they are logical expressions like “If the patient gender is male, then patient must be older than 40”. VISAGET can capture this kind of criteria in a very natural way. In function button panel, we provide a series of buttons and in criteria building panel every widget is selectable and sortable. If you want to build an “If - Then” criterion, just select the two criteria in a proper order, and click the button “2-Group” button, then the two widgets will be put into one group and displayed like the form in figure 8. Similarly we also build If-Then-Else criteria. We call this kind of criteria as group criteria.

(d) Panel for Eligibility Criteria Description — This is directly related to the eligibility criterion on the right. As we discussed in section 3.1, the rendering of placeholder in description is dependent on the value type and specific values of the criterion.

i. Continuous — Taking WBC as an example, let’s assume the selected range is from a to b. Case 1: \(a > 0\) (minimum) and \(b < 100,000\) (maximum), the rendering is “WBC of eligible patients should be between a and b per micro liter”; Case 2: \(a = 0\) and \(b < 100,000\), the rendering is “WBC of eligible patients should be less than b per micro liter”; Case 3: \(a > 0\) and \(b = 100,000\), the rendering is “WBC of eligible patients should be greater than a per micro liter’.’

ii. Boolean — Taking gender as an example, the rendering of description is “Eligible patients should be male” if the selected gender is male.

iii. Categorical and Dynamic Categorical — The rendering of criteria description is dependent on the possible options specified for the criteria. For example, the rendering of description is “Eligible patients’s diagnosis should be one of A or B” if eligible diagnoses are
A and B; If there is only one eligible option A for the criteria, the
description is “Eligible patient’s diagnosis should be A.”

(e) Button to save a single criterion — Any changes to the criterion will
trigger this button to be a star icon representing the status of changes.
Click this button will save the criterion to the related CT.

(f) Button to save all criteria — Click this button, changes to all criteria are
save the CT, which is equivalent to click every save button in each single
criterion.

5.4.6 Matching Algorithm

As designed in Section 3.4, the matching between patient and trial is done through
patient data item and trial eligibility criteria. Figure 5.9 demonstrates the steps of
this matching algorithm.

Starting with a Clinical Trial denoted by T and a patient denoted as P:

1. The first step of this algorithm is fetch all eligibility criteria of T denoted as
   ECs;

2. Decide if the ECs are empty or not;

3. If ECs are empty, go to Step 10, else go to Step 4;

4. Get one eligibility criterion EC from ECs;

5. Get related ECO term of EC;

6. Get the patient data item according to ECO term id and Patient id;

7. Decide if patient data item meets the requirement of EC;

8. If the result of Step 7 is no go to Step 11, else go to Step 9;
9. Remove EC from ECs and go to Step 3;

10. Return result: P is eligible for T, go to Step 12

11. Return result: P is not eligible for T, go to Step 12

12. End of matching.

Figure 5.9: Matching Algorithm
5.5 Pilot Results of VISAGET

As the core component, VISAGET is deployed within Trial Prospector in the Gastrointestinal Oncology subspecialty clinics at Seidman Cancer Center on a pilot basis from December 2012 to June 2013. During this time period, Trial Prospector generated a matching report for each new patient visit that could be reviewed by the physician as part of the regular clinical workflow. A total of 84 Trial Prospector reports were created for 15 active clinical trials that were considered for matching purposes. The patient characteristics were tabulated. Males outnumbered females (54.8% vs. 35.5%) with a median age of 65 years within a range of 10-93 years. Most of the patients had a primary diagnosis of Colon cancer (21.5%) followed by Pancreatic Adenocarcinoma (18.3%). Patient had a low incidence rate for Appendiceal cancer (1%), Metastatic Carcinoma (1%), while the majority of patients did not have metastasis (71%).

5.5.1 Patient Eligibility

Trial Prospector performed 1,367 clinical trials matching with an average of 7 eligible trials identified per patient. In addition, an average of 9 ineligible trials were identified per patient. Figure 5.10 displays the list of conditions that resulted in exclusion of patients from a trial (conditions that did not cause exclusions were not plotted in the chart.) The most common reasons for ineligibility were the pathologic diagnosis and various types of laboratory test results, including total bilirubin (18.3%) and creatinine (11.4%). About 3% of the patients were excluded from trials based on their cancer stage information and also if they had metastatic cancer (2.2%), which highlight the importance of the Trial Prospector UI module feature allowing physician to update these values and generate new match results for a patient.
5.5.2 User Evaluation

Eleven medical oncologists (6 attending physicians and 5 clinical fellows) completed two user satisfaction surveys using REDCap tool [20] after each new patient encounter. Prior to the deployment of Trial Prospector, a survey of the control group was conducted to assess baseline practice characteristics for 14 patient visits. About 57% of the respondents thought of clinical trials as a treatment option and 62.5% of the respondents discussed available clinical trials with patients in the control group. A subsequent survey was conducted after deployment of Trial Prospector covering 60 new patient visits.

In the Trial Prospector group, the treating oncologist considered clinical trials for 66.7% of the patients. 95% of the participating oncologists reviewed the Trial Prospector report at the point of care with 70% spending 0-5 minutes to assess the eligibility of a patient for the available trials. The physicians used the Trial Prospector match report for 72.5% of the patient visits. Moreover, oncologists reported that
Trial Prospector reduced the time taken to identify potential trials during 57.1% of the visits. About 72.7% of the participating oncologists stated that Trial Prospector made it easier to find clinical trials for their patients. Furthermore, 81.8% of the participating physicians would recommend Trial Prospector to other physicians for clinical trial eligibility screening. The physicians reported they liked the ease of use, auto-population of data, and explanation of why the patient was ineligible for certain trials.

Several areas for improvement were also suggested, including use of more eligibility criteria and stricter matching criteria by the matching algorithm. It was also suggested that phase I trials be automatically excluded from the report when the patient is being considered for adjuvant therapy. Overall these results indicate that Trial Prospector is a feasible, accurate, and effective means to identify clinical trials for individual patients in a busy outpatient oncology clinic.

5.5.3 Performance Scalability Evaluation

Trial Prospector is a scalable tool that efficiently computes matching reports over increasing number of both clinical trials and patients. Figure 5.11 shows the performance of Trial Prospector as the number of patients increase from 20 to 80 with 5, 10, and 15 trials. To support larger patient populations and number of trials, we propose to implement the CDE and PSM modules using cloud-computing infrastructure to enhance the performance of both data extraction and matching. A cloud-based implementation using open source Hadoop platform will allow Trial Prospector to take advantage of the inherent parallelizable characteristics of the data extraction and matching process.
5.6 Conclusion and Discussion

VISAGET proposes a computable modeling of eligibility criteria and patient data. It translates matching between patients and clinical trials into a set of matchings between single patient data item to single eligibility criterion and convert that further into SQL queries. VISAGET has been piloted with Trial Prospector since December 2012 and the results demonstrate it is accurate, user friendly, and helpful for physicians to match patients with clinical trials.
Chapter 6

Active Dashboard

Besides clinical trials, patient cohort identification is also needed for clinical decision support. In long-term workflow in clinical care like the one for deep brain stimulation, one frequent question asked by neurologists is “Who is ready for DBS surgery?” Care conference is one step in the workflow of deep brain stimulation.

Keyword based search is the most common way that is widely used in EMR systems to identify patient cohorts. Keyword based search can locate the patients in the system using demographic information like name, medical record number (MRN), or diagnosis. Search by combinations of these demographic information is also not hard to provide. However, keyword based search cannot help with the neurologist’s question above.

A new mechanism is needed in patient care that can tell patient position in their clinical care workflow based on patient data itself. In this chapter, we present such a method called Active Dashboard that supports fast patient cohort identification in clinical workflow as well as other useful functions like data summary and navigation. Active Dashboard is first created and applied in Database for Deep Brain Stimulation (DBSDB).
6.1 Background

Parkinson’s disease (PD) is one of the most common neurodegenerative disorders. It is usually classified by clinical features such as bradykinesia, tremor and rigidity. According to Parkinson’s Disease Foundation, PD is currently affecting one million people in the United States and an estimated four million worldwide. It is estimated to grow substantially in the next 20 years according to a project conducted in the 5 most populous Western Europe nations and the world’s 10 most populous nations. The total number of individuals with PD is expected to be greater than 9 million in 2030 [53].

6.1.1 Parkinson’s Disease Treatment

Medications are available for the treatment of PD such as levodopa, dopamine agonists and monoamine oxidase (MAO)-B inhibitors. They are widely used to alleviate the PD symptoms and reduce the risk of dyskinesia. For the PD patients with major depressive disorder, interpersonal psychotherapy is proved to be a feasible treatment to improve depression symptoms [54]. However, despite the advances of pharmacotherapy and psychotherapy, there are drawbacks to each treatment options. The medications take effect initially, but after five years of treatment a majority of patients no longer benefit from pharmacotherapy and even develop medication related motor complications [55] [56]. Psychotherapy is partially or completely ineffective for some patients [57]. Therefore while treatment options are available for most patients, some PD patients are treatment-resistant and even unresponsive to both psychotherapy and pharmacotherapy [58].
6.1.2 Deep Brain Stimulation Workflow

Deep brain stimulation is a FDA approved neurosurgical treatment for PD patients who cannot benefit from either pharmacotherapy or psychotherapy. It is a clinically proven successful technique that offers treatment for PD with concomitant obsessive compulsive disorder or treatment-resistant depression [59].

Among the 25 centers nationwide, the Movement Disorders Center in the UH is recognized extensively for its treatment of Parkinson’s disease and DBS is one of the most innovative and effective techniques. Figure 6.1 shows the workflow of DBS procedure in UH Movement Disorders Center and the patient data generated in each step. First of all, patient demographic and registration information are recorded along with referring physician information. Afterward, the patient make several clinical visits to identify the PD rating scales and other medical information which will be used to evaluate the DBS qualification of the patient. After that, a multidisciplinary team consisting of movement disorders neurologists, functional neurosurgeons, and neuropsychologists conducts a care conference to determine the applicability and optimal surgical plans regarding the DBS procedure. If qualified, the patient undergo stereotactic imaging procedures such as magnetic resonance imaging (MRI) or computed tomography for the physicians to decide the target brain area to implant the micro-electrodes. During the surgery, micro-electrodes are precisely implanted in some brain areas through a combination of stereotactic and neuroimaging techniques. A subcutaneous external pacemaker which is internal programmable is implanted in the patients’ chest area. After the surgery, programming sessions are scheduled and adjustments on the pacemaker are made with respect to electrode configuration, voltage amplitude, pulse width, and frequency to send electrical currents to the brain for the best outcomes. Also all the benefits and adverse events are recorded to help the physicians make appropriate modifications.
6.2 Methods

In this section, we present the modeling of concepts in DBSDB that is related to Active Dashboard including: clinical visit, data form, and patient state. Clinical visit models patient’s visit to clinical setting capturing information of main purpose, providers, and received treatments. Data forms are generated from clinical visits. Data form models the form used in clinical care to capture information about some aspect of patients. As depicted in Figure 6.1, there are total 36 distinct forms in DBS workflow for various information. Patient state is a new concept in Active Dashboard defined as: a stable milestone position a patient stands in that clinicians need to identify first before they can make decisions for the next step. As we have mentioned before, neurologists ask “Who is ready for surgery?” before they start to schedule patients from surgery. Therefore, “Ready for surgery” can be a patient state.

6.2.1 Data Form

There are various data forms captured in DBSDB. A data form can be used for patient registration that captures patient demographic information and another one can be used for family history. DBS program has six cores: administrative core, clinical core, imaging core, surgery core, neuropsychological core and ad-
verse events core. Each core has a set of data forms with very different schemas. Each form is created as a individual data model in DBSDB.

### 6.2.2 Clinical Visit

Figure 6.2 illustrates the relationship between clinical visit and data forms. The main attributes of clinical visit are:

- **Visit date** is important information that can link different data forms together. Visit date can also be used to calculate patient state, e.g., after 6 month of surgery date, patient should be contacted by neuropsychologist for follow up visit.

- **Provider id** is a user id of a clinician who is providing this clinical visit to patients.

- **Visit Status** – There are three possible statuses for a clinical visit: scheduled, completed, and cancelled.

- **Visit Type** – There are four types of clinical visits in DBSDB: neurologist visit, neurosurgeon visit, neuropsychologist visit, DBS education/medication testing visit.

![Figure 6.2: One clinical visit may generate many data forms](image)

Figure 6.2: One clinical visit may generate many data forms
6.2.3 Patient State

Figure 6.3 depicts patient state structure and its relationship to patient model. One patient may have many states in DBS workflow. Details about the fields are described below:

- Name intuitively tells us about what this state is about, e.g., “Ready for surgery” tells us this patient is ready for the scheduling of a surgery.
- Description tells details about a state.
- Short name is usually the capitalization of the first letter. Short name makes patient state fit into widget where space is limited.
- Ordering decides how to order to present multiple states of a patient. States of lower ordering are placed in the front.
- State – We need to emphasize this attribute of patient state. State is an integer of 3 possible values: 0, 1, -1. We describe details about state and state transfer in the following subsection.

![Patient State Diagram](image)

*Figure 6.3: Relationship between patient and patient states: one patient may have many patient states*
6.2.4 Patient State Transfer

As mentioned above, a patient state has three possible values: 0, 1, -1.

- 0 – State is scheduled or possibly true. We assign yellow as the color code for state 0.
- 1 – State is completed or true. We assign green as the color code for state 1.
- -1 – State is not scheduled or false. We assign red as the color code for state -1.

Transfers between states are illustrated in Figure 6.4

Transfer from 0 to 1 means one patient state changes from scheduled to completed. This transfer happens to most patient states.

Transfer between 1 to -1 happens only for patient state that describes something is true or false. For example, patient is ready for surgery at time point 1 and become not ready at a later time point 2 due to medication or other problems.

Transfer from -1 to 0 means a patient state changes from not scheduled to scheduled or from false to possibly true.

I would like to use another patient state “Interested in DBS” to illustrate possible state transfers. At the beginning of DBS workflow, patients usually state that they are possibly interested in DBS. Therefore the state of “Interested in DBS” is 0 (possibly true or possibly interested); Some time later, after patients get further
in DBS workflow they may become explicitly interested in DBS. Then the state for “Interested in DBS” is 1 (true or interested); Finally, patients have DBS surgery operated and they will not have another DBS surgery in short period. Then their state of interest in DBS will change to -1 (false or not interested).

6.2.5 Architecture of Active Dashboard

6.5 depicts the architecture of Active Dashboard. In Active Dashboard, patient data is made up of a number of data forms that is inter-linked by clinical visits. A set of patient states are defined to represent patient positions in DBS workflow. These states are triggered directly by data forms. With these states, Active Dashboard can figure out the patient positions in the workflow and complete the identification of patient cohort. In next sub section we talk about some best practices in DBSDB for data capture which affects the generation of patient states.

6.2.6 Best Practices in Database for Deep Brain Stimulation

Data quality is one of the big concerns of paper-based and document based way of data storage. The following sections describe the techniques and best practices developed in DBSDB data capture interfaces, which largely prevent the manual errors caused by heavy workload of data entry, reduce the need for expensive “post-entry” data curation, and ensure the data quality.

Choose over Type

Patient diagnosis is an free structured text field In the existing EMR system of DBS program and system user needs to manually type the name of diagnosis. There are 410 diagnoses for movement disorders to the best knowledge of our team and these diagnoses form a hierarchical structure. There is a big chance for typos and
inconsistency to occur when manually type diagnosis information. DBSDB builds a library for diagnoses 6.6 for movement disorders and render them in a tree structure. Users of DBSDB just need to choose from the tree structure instead of typing manually. This method eliminates the possibility for typos and inconsistency.

**Providing Default Values**

Data entry in a clinical system can be tedious. For example, one programming setting for macro-electrode in DB2S has more than 10 parameters, including four
contacts setting, mode (current or voltage) setting, pulse width, stimulation frequency and so on. Empirically, DBS patients have about 8 programming sessions after surgery which in total generate more than 250 programming settings. If no assistance provided, a system user need to enter more than 2500 values only for one patient programming data. You can imagine how easily that can introduce manual errors. In DB2S, however, we provide default values for data entry whenever possible. Taking programming data as example again, last programming settings are copied over as default values when a new programming setting is entered because adjacent programming settings of the same macro electrode differ only by one or two parameters. Although the new programing setting has more than 10 parameters to enter, user of DB2S only need to focus on one or two fields that are changed, which significantly reduce the workload for this kind of continuous data entry.
Interactive Data Validation

DBSDB provides 2 general validations for data that is not appropriate for library building like numeric value or free structured comment text. Data validation is important to ensure that data is valid and properly structured before it is sent to database for storage.

- Tooltip or Placeholder — A tooltip or placeholder is provided for a input field in DBSDB to explain what this field is and what should be entered. For example, a placeholder “0-250Hz” in frequency field of programming settings gives the user who enters data the instruction about the data range tolerated here.

- Real-time Feedback — DBSDB uses jQuery javascript library to provide real-time feedback for data entry. It performs presence check for required fields, range check for numeric values, and confirmation check if existing data is overridden.

6.3 Results

In this section, we present the results of Active Dashboard in DBSDB.

6.3.1 Patient States Library

We create 13 patient states for DBSDB representing 13 different stages in the DBS workflow. Details are described in table 6.1.

6.3.2 Active Dashboard User Interface

In DBSDB, we create a concise patient search interface. The patient search interface uses the 13 patient status boxes as template and the configuration of the selected
<table>
<thead>
<tr>
<th>Name</th>
<th>Short Name</th>
<th>Ordering</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interested In DBS</td>
<td>I</td>
<td>1</td>
<td>Patient is interested in DBS</td>
</tr>
<tr>
<td>Neurologist Visit</td>
<td>N</td>
<td>2</td>
<td>Neurologist Visit is done</td>
</tr>
<tr>
<td>DBS Neurologist Visit</td>
<td>D</td>
<td>3</td>
<td>DBS Neurologist Visit</td>
</tr>
<tr>
<td>Neurology Nurse Visit</td>
<td>Nn</td>
<td>4</td>
<td>Neurology Nurse Visit is done</td>
</tr>
<tr>
<td>Neurosurgery Visit</td>
<td>Ns</td>
<td>5</td>
<td>Neurosurgery Visit is done</td>
</tr>
<tr>
<td>MRI</td>
<td>M</td>
<td>6</td>
<td>MRI is done</td>
</tr>
<tr>
<td>Neuropsychologist Visit</td>
<td>Np</td>
<td>7</td>
<td>Neuropsychologist Visit</td>
</tr>
<tr>
<td>Care Conference</td>
<td>C</td>
<td>8</td>
<td>Care Conference Held</td>
</tr>
<tr>
<td>Ready for Surgery</td>
<td>R</td>
<td>9</td>
<td>Patient is Ready for Surgery</td>
</tr>
<tr>
<td>Surgery</td>
<td>S</td>
<td>10</td>
<td>Surgery is done</td>
</tr>
<tr>
<td>Programming</td>
<td>P</td>
<td>11</td>
<td>Electrodes are programmed</td>
</tr>
<tr>
<td>Neuropsych Post-Op</td>
<td>Po</td>
<td>12</td>
<td>Neuropsych Post-Op is done</td>
</tr>
<tr>
<td>Post-Op MRI</td>
<td>pM</td>
<td>13</td>
<td>Post-Op MRI is done</td>
</tr>
</tbody>
</table>

Table 6.1: Patient states of DBSDB

colors for the intended results. One can click to change individual status box to red, green, or white.

The coloring of the status box reflects its possible state: green indicates “completed” or “confirmed,” red means “incomplete” or “negative,” and white represents “unknown.” Status change is automatically triggered during data entry. For example, the I status (interested in DBS) of a patient is changed from white to green once a registration form is entered, in which the field “interested in DBS” in registration form is selected as “Yes.”

Figure 6.7: Query interface of active dashboard

Patient statuses are displayed on the left. Mouse-over the status area will trig-
ger the display of the full name of each patient state. With patient status information stored in DB2S, this patient search interface can find all patients satisfying a certain criteria expressed by the query template. For example, to answer the question “Who are ready for care conference,” clinicians just need to perform 7 clicks on all status boxes in the template before C to make them green, two clicks on C to make it red, leaving the status for the rest of the boxes as is (white).

The patient search interface returns a list of patients satisfying the conditions specified in the template. Each patient record has its status array displayed in its record row. Each box in the status array provides additional information about the status for the specific patient via mouse-over. As shown on the bottom left of Figure 6.7, when mouse hovering over status box D, it shows DBS neurologist visit was completed on 12/03/2014. In addition, a user can click the patient status box to display the relevant data form. As illustrated in Figure 6.7, clicking on the status box D leads to clinical visit form on 12/03/2014 of the patient (right of Figure 6.7).

6.4 Evaluation

Deployed in April 2014, DB2S captures 236 patient records in less than one year. Clinicians rely on it not only to capture patients data in the DBS program of UH Movement Disorders Center, but also document previous patients who have already completed the procedure. Using DB2S patient search interface, we get the statistics about the patients captured in the system displayed in Figure 6.8 using queries that are most useful in the DBS program. Query 1 and 2 found 23 of 236 who are interested in DBS and 39 who are not interested while the other 174 do not clearly indicate their interests. Further inspection of patient data shows that 172 of those 174 patients have already completed the DBS procedure. Patient status usually turns into “not interested” in DBS after operation is done. Query 4
indicated that 38 of the 39 patients who are not interested in DBS is because of the completion of their surgeries. Query 11 found patients with care conference done but still not ready for surgery. The rest of the queries are frequently used for finding patients that are ready for some clinical procedure or visit such as neurologist visit, care conference, or MRI, but those procedures or visits are not scheduled. All such query results are easily validated using the direct links to the respective data forms provided in the resulting status arrays.

<table>
<thead>
<tr>
<th>#</th>
<th>Sample Clinical Query</th>
<th>Status Template</th>
<th># of Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>All patients in DB2S</td>
<td>INDNnNsnMnPCRSPPoPM</td>
<td>236</td>
</tr>
<tr>
<td>2</td>
<td>Patients that are interested in DBS procedure</td>
<td>INDNnNsnMnPCRSPPoPM</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>Patients are not interested in DBS procedure</td>
<td>INDNnNsnMnPCRSPPoPM</td>
<td>39</td>
</tr>
<tr>
<td>4</td>
<td>Patients that have surgery done and become not interested in DBS</td>
<td>INDNnNsnMnPCRSPPoPM</td>
<td>38</td>
</tr>
<tr>
<td>5</td>
<td>Patients that need be scheduled for neurologist visit</td>
<td>INDNnNnsMnPCRSPPoPM</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>Patients that need be scheduled for neurology nurse visit</td>
<td>INDNnNnsMnPCRSPPoPM</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>Patients that need be scheduled for neurosurgery visit</td>
<td>INDNnNnsMnPCRSPPoPM</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>Patients that need to do MRI Imaging</td>
<td>INDNnNsnMnPCRSPPoPM</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>Patients that are ready for neuropsychology</td>
<td>INDNnNsnMnPCRSPPoPM</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>Patients that are ready for care conference</td>
<td>INDNnNsnMnPCRSPPoPM</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>Patients that have care conference but not ready for surgery</td>
<td>INDNnNsnMnPCRSPPoPM</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>Patients that are ready to have DBS operation</td>
<td>INDNnNsnMnPCRSPPoPM</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>Patients have surgery done but programming is not scheduled</td>
<td>INDNnNsnMnPCRSPPoPM</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>Patients that need post-op neuropsychology</td>
<td>INDNnNsnMnPCRSPPoPM</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>Patients that need post-op MRI imaging</td>
<td>INDNnNsnMnPCRSPPoPM</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 6.8: Statistics about patient statuses captured in DB2S
6.5 Conclusion and Discussion

In this chapter, we present Active Dashboard, a novel data organization and patient cohort identification method. Applying Active Dashboard in DBSDB, we create 13 patient states to represent different stages of patients in DBS workflow. Patient states are triggered automatically by data forms generated from clinical visits. We also create an interactive search interface to facilitate fast patient cohort identification by patient states. Neurologists can fast identify patient cohorts of interest within a few mouse clicks. In addition to cohort identification, Active Dashboard also supports patient information summary and direct data navigation.
Chapter 7

Web Interface Driven Development

Agile methods of software development has become very popular in recent years [19, 20, 22, 21]. Iterative and incremental development, evolving since 1950s, has taken the place of waterfall model as the main-stream style of software development [19].

A large number of agile methodologies have emerged such as extreme programming (XP), feature driven development (FDD), the rational unified process (RUP), and so on. Unlike traditional methodologies like waterfall model being too mechanistic to use in practice, agile methods put more emphasis on changes that involving people, interaction, working software and client collaboration [20]. In this chapter, we introduce a novel IID method called web interface driven development specially for clinical data management system.

7.1 Motivation

A typical lifecycle of IID consists of requirements collection, analysis and design, implementation, testing, evaluation, deployment, and maintenance [21]. Of the lifecycle, requirements collection is one of the most well-known places we have to revise our work again and again. Researches have been done about how to
integrate traditional software engineering methods with agile software development [23, 24, 25, 26, 27]. There is no easy simple answer to the question of which is better: traditional requirement engineering or agile methods [23]. [24] states that unlike traditional requirement engineering relying on documents to guide developers to do that right thing, agile methods put more emphasis on skilled developers to do the right thing. Also it argues that agile methods should have at least a minimum of documentation for maintenance purpose. The common understanding is agile methods focus on delivering critical functionalities to users and solve requirements engineering by continuously involving customer interactions throughout the whole development life cycles. However, these studies don’t give us clues about how agile methods can solve the key challenge of requirements: communication problems [28, 29].

Information systems have always been developed through social processes, especially for requirements collection. In the process, actors playing a variety of specialized roles like client, developer, and project manager interact and collaborate to produce new applications. Understanding client, their needs and how they will apply the proposed system to better complete their work and achieve their expectations can greatly increase the likelihood of successful projects [29].

Requirements collection is difficult because it relies on domain knowledge, which is especially the case for clinical data management system. It requires years of training to be able to work in the clinical settings. Ideally, there will be no communication obstacles if every member of the team is domain experts. But it is not realistic to expect a software developer to have knowledge about procedures in Deep Brain Stimulation. On the other hand, doctors or fellows are not likely to understand database design or what Model View Controller (MVC) pattern is.

The good thing is we find from our years of development experience that our clients can easily tell us they like it or not when we demonstrate them our product.
They have clear expectations about the interface they will work with although they cannot describe that in a way that our developers can understand until they see the product.

Therefore, we create Web Interface Driven Development (WIDD), a new member of the agile methods family, as the methodology for the design and implementation of all our clinical data management systems. In WIDD, web interfaces serve as the common medium that clients and developers can both understand without deep domain knowledge. WIDD does not cover the entire software development, but rather focuses on the requirements collection, analysis and design, implementation, and evaluation. In the following sections we will use development of DBSDB system to exemplify the WIDD workflows and concepts. In the following sections, we will use interface and web interface alternatively since our systems are all web applications.

7.2 Roles and Responsibilities

As mentioned earlier, development of every information system is social processes that involves many actors to play different roles. WIDD classifies 6 roles in its processes into 2 categories: core role and supporting roles. Core roles include project manager, chief architect, lead developer, developers, and domain experts. One supporting role is meeting coordinator.

7.2.1 Project Manager

Project manager is accountable for the success or failure of the project. Specific responsibilities of a project manager include:

- Project Planning — defining the project scope, building comprehensive work plan, and managing to the budget.
• Team Management - facilitating commitment and productivity, and motivating team members.

• Progress Management - preparing progress report, aligning project progress with timelines, and communicating unexpected obstacles or difficulties.

• Resource Management - Allocating resources including equipment, space, and programming resource.

7.2.2 Chief Architect

Chief Architect is in charge of the overall system design. He decides the framework selection, system architecture, implementation strategy, performance control, and scalability expectation. These responsibilities require chief architect to have both excellent communication and technical skills, as well as deep understanding of domain knowledge. Chief Architect performs initial communications with clients to achieve agreements on the expectations and final deliverables of the project.

7.2.3 Lead Developer

Lead Developer is responsible for leading daily development activities. Lead Developer need to have excellent technical skills including design and modeling skills. Lead developer should have experience of how to write and maintain sustainable codes and how to avoid potential performance and scalability pitfalls. Typical responsibilities of Lead developer include:

• Participating in weekly meeting with project manager, chief architect, and clients or domain experts to map;

• High level design from requirements;

• Extracting development tasks from high level design;
• Estimating timelines for each development task;
• Assigning development tasks to developers;
• Holding regular code review meetings to follow up with progress and resolve possible technical obstacles.

7.2.4 Developers
Developers work team member under the guidance and instruction of Lead developer to work on part, or one, or multiple interfaces. Developers are responsible for the detail design, implementation, test, and documentation tasks of the interfaces they are working on.

7.2.5 Domain Experts
In WIDD, domain experts are exactly our clients and users. Domain experts are the people who will really use our system to facilitate their regular work to enable more possibilities or achieve better performance or results. For example, in DBSDB project, Dr. Benjamin Walter is our client, who is a Neurologist and the medical Director of Deep Brain Stimulation Program of UH Case Medical Center. He is right now using this system for his daily work. He is the main person who is communicating with us about requirements.

7.2.6 Meeting Coordinator
In WIDD, we need only one supporting role: meeting coordinator. Meeting Coordinator is responsible to coordinate the schedules of all roles to hold regular weekly or bi-weekly meetings. It is also part of the duties of meeting coordinator to remind people of any changes of time and place of regular meeting and keep record of every meeting note.
7.3 Processes

WIDD is an iterative, agile web application development specially designed for clinical data management system. It applies interactive web interface as the communication medium to collect requirements and perform review sessions iteratively to allow clients to adjust their expectations. WIDD has three sequential processes illustrated in figure 7.1.

7.3.1 Build Interfaces List

As WIDD begins, domain experts (Dr. Walter in DBS) and project manager should have already have some initial meetings talking about the project scope and overall expectations.

Based on the agreement on that, Domain experts will walk through the team all functionalities they expect. During this process, domain experts can use existing system or screenshots to help development team to better understand them.

The whole team brainstorms about the requirements and chief architect decides the system architecture and development strategy.

Lead developer and domain experts then work together to build a list of interfaces. Each interface should solve one or more functionality expectations and all of them should cover all the functionalities.
7.3.2 Plan by Interface

After the list of interfaces is built, lead developer need to make development plan for each of the interface.

Lead developer needs to

- Divide one interface into workable small pieces considering related data models;
- Assign development tasks to developers and estimate the working timeline;
- Notify meeting coordinator to schedule regular development meetings with domain experts.

7.3.3 Interface Lifecycle

Implementation begins after the interface plan is made. There are five critical steps in the implementation lifecycle: requirement details collection, data model design and initial interface implementation, review and feedback, revise, implementation of model and controller. Figure 7.2 illustrates the five critical steps in web interface driven development.

7.4 Interface Lifecycle

7.4.1 Requirements Details Collection

Developers need to well understand the clinical background knowledge before they can make design decisions about their assignments. However, software developers usually have no clinical background. Therefore, it is necessary to put requirement details collection as the first step of the lifecycle of interface implementation.
We rely on our clients to provide more detailed information to specific interface to make it ready for implementation. The information can be in the following formats.

- Screenshot – Domain experts can provide screenshots from existing EMR system if that exists. Screenshots are very helpful information that they accurately state a set of related data elements, their data types, and their in-
ner relationships. Based on the screenshots, domain experts can describe the changes to be made about the existing interface.

- **Formal Document** – Usually there are forms in PDF or other data format out there for electronic web forms we need to build. These forms formally describe the form structures, all the fields we need to include and all the possible values of each field. For example, Movement Disorder Society-Sponsored Revision of the Unified Parkinson’s Disease Rating Scale (MDS-UPDRS) [35] tells us there are four sections in the form, total 75 questions in these sections, and all the details about these 75 questions. In addition to the forms, domain experts can ask more fields to be added like sum score of each section, sum score of all fields and so on.

- **Spreadsheet** – Domain experts can provide data in spreadsheets to us if no screenshot or formal document is available. Spreadsheet is good at showing data structure and hierarchy.

- **Whiteboard** – Clients can also write and draw on the whiteboard to express their concepts, ideas, and requirements. Our developers will take pictures of the whiteboard and keep those in the project document for future discussion and maintenance.

### 7.4.2 Initial Interface Building

Developers can start building the interface after they acquire enough information from previous step.

At first, developer need to design the data model behind the interface which define the type and possible values of each data element that will be finally built on the web pages. To be clear, at this step, no implementation about data models is
needed, which means no data tables will be really created because this data model is temporary and will experience a lot of adjustments in the future development.

After finishing data model design, the implementation of interface is straightforward. Developers can choose the best way of representation to the best of their knowledge. For example, for MDS-UPDRS, a developer chooses accordion style over tab style because MDS-UPDRS section name is fairly long and that will make tab titles look unpleasant.

We do not implement controller logic at this point either since no data table is implemented until this step. However, we do need interactions between users and our web interfaces. JavaScript is our choice for interaction. JavaScript can simulate the same effects after all data tables and business logic are created and give users the right feedback for their actions.

7.4.3 Review Session

Once a web interface is completed, a review meeting is conducted with both developers and clients. In the meeting, the web interface is demonstrated. Both developers and domain experts are on the same “pages” (web pages). Clients can play with the interface lively and point out what field is missing or what feature is not as expected with no difficulty. Developers document the feedback and share the documentation with meeting coordinator.

7.4.4 Interface Revision

In this step, developers revise their previous work to satisfy the new requirements reflected in the feedback collected from previous review session. The revise work including three aspects: data model design change, interface change, and JavaScript interaction change.
7.4.5 Implementation of Model and Controller

Review and revise is an inner loop in the interface lifecycle. Although the loop can happen several times until no changes are needed, it usually ends very fast because the communication barrier is avoided. Developers can react fast to the feedback from users and the amount of changes decrease dramatically after each review session. The average number of loops is 2 to 3 according to our project experience.

After the last review session, clients are fully satisfied with built interface and interactions. Developers can create data tables using the revised data model design and implement the business logic by mapping the JavaScript simulation to database operations.

7.5 Practice

In this section, I will use programming interface of DBSDB as an example to demonstrate the steps of WIDD Processes.

Deep brain stimulation implants electrodes into patient’s brain. The electrodes are programmed to send pulses to achieve the treatment purpose. The programming interface we are trying to build is a place to record all the programming settings about a patient’s electrodes. Together with the programming settings we also need to store the outcome of each setting.

7.5.1 Requirements Details Collection

We collect the following requirements from the clients about the programming interface.

1. Programming interface needs to include the following information: program-
ming date and time, electrode information, settings of contacts of the electrode, current or voltage mode, pulse width, frequency, outcomes including both benefits and side effects.

2. There are one case and four contacts (0, 1, 2, and 3) in one electrode. Value of case and contact can be positive or negative or not used.

3. Pulse width is between 0 and 450 micro seconds and frequency is between 0 to 250 HZ. Benefit and side effect outcomes are free text descriptions.

4. The interface should support search function by programming date or lead.

### 7.5.2 Initial Interface Building

Figure 7.3 shows the initial data modeling related to programming interface.

![Data Modeling](image)

**Figure 7.3: Initial data modeling related to programming interface**

Figure 7.4 is the overview of the programming records. It is a single table view design. Each row is a programming record with attributes from left to right as: programming date and time, contact setting, programming mode (current or voltage), pulse width, frequency, outcome, and operation buttons for edit, delete and open other information.

Figure 7.5 shows the edit page of the a programming record. We can see that widget choices for different columns:
Figure 7.4: Initial building of overview page in programming interface

- Date picker from jQuery is used for date selection. Two dropdown lists are for hour and minute selections;
- Contact setting has one dropdown list for each of the contacts and a number field for the value of voltage or current depending on the mode setting;
- In the dropdown selection for mode, “C” represents current mode and “V” represents voltage mode;
- Pulse width and frequency are two number fields;
- In outcome column, two free text fields are for benefit outcome and side effect outcome respectively.

Figure 7.5: Initial building of edit page of a programming record

7.5.3 First Review Session

After looking at the demo of initial interface, clients find out the following aspects need to be addressed in the future iteration:
1. Programming records should be grouped as separate sessions. Programming session should be created first and then programming records are added into the session. A programming session should have date and lead information. Programming record should have time and other remaining information.

2. One programming can have up to two groups of contacts. The two groups of contacts have consistent mode (both current or both voltage) but with independent values.

3. Outcome should include structured benefits and side effects. Benefit and side effect have same schema but with different value options for the name field. Possible names for benefit are: Tremor, Rigidity, Bradykinesia, and Dystonia. Possible names for side effect are: Paresthesia, Capsule, Oculomotor, and Optic.

### 7.5.4 First Revision

Figure 7.6 shows the revised data modeling that addresses the new requirements from review session No. 1. Three new models are added: Programming Session, Side Effect, and Benefit. Contact model is renamed as Programming Group to make more sense.

Figure 7.7 is the revised overview of the programming records. The overview page is changed to multiple tables with each programming session as a table. On top of each table view is the summary information about that programming session including programming date and programming lead information. Contact setting column is expanded to be able to show two groups of settings.

Figure 7.8 shows the edit page of the a programming record. The main changes showing in the figure are two programming groups and the benefit and side effect buttons. Under save button, there is a ”+group” or ”-group” button to add a group
Figure 7.6: Revised data modeling related to programming interface

Figure 7.7: Revised overview page in programming interface

to the programming or remove a group to the programming. Figure 7.9 shows one example of structured benefit configuration while Figure 7.10 shows one example of structured side effect configuration.
7.5.5 Second Review Session

Revised work greatly increase the satisfaction of our clients. After reviewing the functions of the added features, they raise two more adjustments:

1. Programming session can hide or expand. Hiding a session will hide all intermediate programming records and expanding a programming session will show all programming records in a session.

2. Programming session can be finalized and finalized session cannot add any more programming record.
7.5.6 Second Revision

For data models, in this iteration we just need to add one boolean field “finalized” to Programming Session model.

For view, two buttons are added on the top right of programming session summary banner illustrated in Figure 7.11. “Finalize” button is to prevent current programming session from adding new records. “Hide” button is to hide all intermediate programming rows except the first and last programming.
7.5.7 Third Review Session

At the third review session, our clients think this programming interface satisfies all their needs. Then we can continue to work on implementation of data models and controllers.

7.5.8 Implementation of Model and Controller

This step is straight forward. We can create finalized data tables according to the data model finalized after review session 2. Then for controllers, we modify the actions to read from and write to these real data tables. After that, the implementation of programming functionality is complete. Next step is the following work in traditional iterative development like testing and deployment.

7.6 Development Environment

Table 7.1 illustrates the system environment we use for WIDD.

<table>
<thead>
<tr>
<th>Name</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ruby</td>
<td>1.9.3</td>
</tr>
<tr>
<td>Rails</td>
<td>3.2.13</td>
</tr>
<tr>
<td>MySQL</td>
<td>5.5.x</td>
</tr>
<tr>
<td>jQuery and UI</td>
<td>1.9.x</td>
</tr>
<tr>
<td>Bootstrap</td>
<td>2.3.x</td>
</tr>
</tbody>
</table>

*Table 7.1: VISAGET development environment*

7.6.1 Ruby on Rails

Ruby on Rails is an open-source agile web framework. It follows MVC (Model-View-Controller) pattern to separate work into models, views, and controllers. This design provides great supports for WIDD.
7.6.2 jQuery and jQuery UI

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. jQuery UI provides a set of user interface interactions, effects, and widgets on top of jQuery, which is a great fit for WIDD interactive interface before data models and controllers are implemented.

7.6.3 Bootstrap

Bootstrap is a powerful front-end framework for faster and easier web development. It saves us the time to design page layout and html elements styles.

7.6.4 MySQL

MySQL is a very popular open-source database management system. It provides stable performance and easy to use APIs. Besides Ruby on Rails provides perfect support for MySQL usage.

7.7 Results

In DBSDB, we create 36 distinct data forms. These forms contain 1,109 data fields (see Table 7.2). Each data field captures essential information about patient information. Following WIDD agile methodology, we created at least two types web interfaces for these forms – one for data entry and one for data display. In total, DB2S has more than 72 distinct web pages for patient data capture and display.
## Table 7.2: Distinct forms in DBSDB and the number of data fields captured in them

<table>
<thead>
<tr>
<th>Form Name</th>
<th>No. of Data Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demographic</td>
<td>30</td>
</tr>
<tr>
<td>Registration</td>
<td>19</td>
</tr>
<tr>
<td>Diagnosis</td>
<td>15</td>
</tr>
<tr>
<td>Genetics</td>
<td>10</td>
</tr>
<tr>
<td>DNA Test</td>
<td>5</td>
</tr>
<tr>
<td>Family History</td>
<td>13</td>
</tr>
<tr>
<td>Initial Medications</td>
<td>10</td>
</tr>
<tr>
<td>Final Medications</td>
<td>10</td>
</tr>
<tr>
<td>UPDRS</td>
<td>76</td>
</tr>
<tr>
<td>MDS–UPDRS</td>
<td>60</td>
</tr>
<tr>
<td>TRS</td>
<td>56</td>
</tr>
<tr>
<td>BFMDRS</td>
<td>45</td>
</tr>
<tr>
<td>TWSTRS</td>
<td>36</td>
</tr>
<tr>
<td>UDRS</td>
<td>47</td>
</tr>
<tr>
<td>Epworth</td>
<td>17</td>
</tr>
<tr>
<td>MMSE</td>
<td>20</td>
</tr>
<tr>
<td>MoCA</td>
<td>46</td>
</tr>
<tr>
<td>SF36</td>
<td>41</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Form Name</th>
<th>No. of Data Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBS Neuropsychology</td>
<td>127</td>
</tr>
<tr>
<td>Post-Op Imaging</td>
<td>13</td>
</tr>
<tr>
<td>Care Conference</td>
<td>28</td>
</tr>
<tr>
<td>Procedure Plan</td>
<td>5</td>
</tr>
<tr>
<td>Surgery</td>
<td>20</td>
</tr>
<tr>
<td>Microelectrode</td>
<td>19</td>
</tr>
<tr>
<td>Macroelectrode</td>
<td>22</td>
</tr>
<tr>
<td>Monopolar Review</td>
<td>17</td>
</tr>
<tr>
<td>Programming</td>
<td>45</td>
</tr>
<tr>
<td>Battery and Lead Impedance</td>
<td>32</td>
</tr>
<tr>
<td>Therapy Impedance</td>
<td>7</td>
</tr>
<tr>
<td>OR Contact Threshold</td>
<td>10</td>
</tr>
<tr>
<td>IPG</td>
<td>11</td>
</tr>
<tr>
<td>Record Site</td>
<td>14</td>
</tr>
<tr>
<td>Micro/Macro Stimulation Site</td>
<td>26</td>
</tr>
<tr>
<td>Post-Op Neuropsych</td>
<td>127</td>
</tr>
<tr>
<td>Post-Op Imaging</td>
<td>13</td>
</tr>
<tr>
<td>Adverse Event</td>
<td>17</td>
</tr>
</tbody>
</table>

### 7.8 Conclusion and Discussion

In this chapter, we present a novel iterative agile software development method called Web Interface Driven Development or WIDD.

In typical iterative incremental development, data models and controllers are the most frequent places that developers need to revise their work because of the difficulty of collecting complete user requirements at early iterations. Empirically, revising data models takes a lot of efforts especially when data migration is needed. To avoid this problem, WIDD successfully defers the implementation of data models and controllers to the last step after all requirements are collected. To achieve this goal, we create interfaces using web pages to bridge the communication between developers and clients. Interactions between clients and interfaces are simulated by javascript. Interface Lifecycle consisting of work, review, revise is created for accumulatively collect user requirements. At the end of last interface
lifecycle, clients are satisfied with the interface design and data model design and business logic design are also finalized during the process.

7.8.1 Applicability

WIDD proves to be an effective method for the development of clinical data management systems like Trial Prospector, DBSDB, and OPIC. However, this methodology is not applicable to all applications.

The key idea of WIDD is to use interface as the communication channel between clients and developers to effectively collect requirements. WIDD works only if clients can precisely reflect end users’ needs and expectations. Otherwise, revise work is inevitable after deployment if clients and users have different understandings about the system. In the development of Clinical data management systems, we do not have this concern because clients, domain experts, and end users are the same group of people.
Chapter 8

Conclusion

In this dissertation, we propose a general architecture design of clinical data management system to address the common challenges existing in various clinical settings such as: how to capture patient data accurately and consistently, how to support secondary use of health data, how to reduce dependency of manual work in clinical workflow, and how to provide efficient patient cohort identification mechanisms.

First of all, we present our work on a generic configurable role based access control functionality module. We build different levels of access control mechanisms in cRBAC based on the real needs of different applications. Correctly configured, this module can properly manage users to perform operations only allowed by the system and focus on their own work and responsibilities. Therefore, cRBAC can avoid many security issues as well as improve working efficiency by helping users focus on the information they are supposed to see.

Followed by that, we talk about a design methodology called ontology guided data capture. From ontological terminology role, data model design and mapping, interactive interface design these three aspects, we use OPIC as an example to present how to apply ontology into data capture systems. Ontology acts
as a knowledge base providing data standardization for OPIC. We map ontology
terms into data models and build helper widget to assist data entry for end users
to avoid possible manual errors and achieve accuracy, consistency, and high effi-
ciency in clinical data capture. The well structured data provides strong support
for secondary use.

Then we propose two different mechanisms of patient cohort identification:
VISAGET and Active Dashboard.

VISAGET aims to provide automation support for screening process that re-
cruit patients to clinical trials. VISAGET creates a computable modeling of clinical
trial eligibility criteria and an eligibility criteria library from NCI common data
elements. It provides an interactive interface to facilitate trial managers to build
eligibility criteria. VISAGET then can translate matching between patients and
clinical trials to patient data items to computable eligibility criteria.

Active Dashboard is designed to fast identify patient cohort in a complex clini-
cal workflow. Active Dashboard defines patient state to represent a stage of patient
in clinical workflow. A system may have a lot of such patient states according to
real needs. These states are automatically triggered by patient data forms that are
captured by the system where Active Dashboard is deployed. Active Dashboard
provides an interactive search interface to assist clinicians to build queries based
on patient states. The widget design is so concise and elegant that the query build-
ing process is intuitive and fast. Patient cohort identification tasks that may take
hours or days of work now can be done within a few mouse clicks. In addition,
active dashboard also provides information summary for each patient state and
direct navigation to the patient data form that triggers that patient state.

At last, we propose a novel iterative agile software development method called
web interface driven development. This method is applicable to projects where
clients, domain experts, and users are the same group of people. The key WIDD
is to implement interface and use interface to bridge the communication between clients and developers and defer the implementation of data models and controller actions after interface is finalized. Interface is a number of web pages and interactions between user and web pages can be simulated by javascript. Interface review can help developers incrementally collect user requirements. Clients and developers work together iteratively to finalize the interface and interactions. After that, data models and controller actions are also finalized and ready for implementation. WIDD avoid the revise work on data models and controller actions by focusing on interface implementation first.

8.1 Deployment in Production Systems

The approaches proposed in this dissertation are all applied in live production clinical systems: Trial Prospector, OPIC, and DBSDB.

Trial Prospector is deployed at point of care to automate the matching process between patients and clinical trials. It uses cRBAC to manage user access. With VISAGET as the matching engine, it provides a workflow that oncologists can identify eligible patients for target clinical trials in a couple of minutes.

OPIC is created and deployed in UH epilepsy center for clinical use. It is also a tool for CSR and deployed in 7 institutions of CSR for research purpose. It also adopts cRBAC and ontology guided data capture method ensures it can capture accurate, consistent, and well structured data that can improve quality of clinical care and also make it possible for large scale, multi-institutional study like CSR to collect interpretable data from different sites.

DBSDB manages complex workflow in deep brain stimulation program with ontology guided data capture and Active Dashboard. As a multi-core program, cRBAC allows it to manage user access easily. Active Dashboard greatly reduce
the time and efforts required to identify patient cohort in the DBS workflow.

8.2 Future Work

There are many aspects to improve in the work of this dissertation. We will focus on three aspects in the future.

8.2.1 Improve Proposed Methods

VISAGET currently is only for cancer clinical trials and can not build eligibility criteria that contain temporal information. One important future work is to expand its ability to build temporal eligibility criteria. Another work is to build a eligibility criteria library that is applicable to general clinical trials.

Active Dashboard is a really useful tool. One interesting future work is to generalize it to other projects like OPIC evaluate the completeness of patient data captured. We can treat each data section as patient state and identify patient cohort based on the combination of data completeness of all data sections.

8.2.2 Propose New Methods

Our architecture for clinical data management system is not perfect. We don’t have a systematic visualization method for all kinds of data we capture. We also don’t have a systematic way to audit all system operations. We will put efforts on these two problems in the future.
Appendix A

Epileptic Seizure Terminologies
<table>
<thead>
<tr>
<th>EPILEPTIC SEIZURE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>AURA*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Somatosensory aura*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visual aura*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Auditory aura*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Olfactory aura*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Gustatory aura</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Autonomic aura</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tachycardic aura</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Abdominal aura</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vasomotor aura*</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pilomotor aura*</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Sialorrheic aura</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sexual aura</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Urinary urge aura</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Psychic aura</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phobic aura</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Experiential aura</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUTONOMIC SEIZURE</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tachycardic seizure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bradycardic seizure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Abdominal seizure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Emetic seizure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vasomotor seizure*</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pilomotor seizure*</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hyperphoric seizure*</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sialorrheic seizure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sexual seizure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Urinary incontinent seizure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DYSCOGNITIVE SEIZURE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amnestic Seizure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apraxic seizure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aphasic seizure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dialeptic seizure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOTOR SEIZURE*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIMPLE MOTOR SEIZURE*</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Epileptic Spasm*</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tonic Seizure*</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clonic Seizure*</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tonic-clonic seizure</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Myoclonic Seizure*</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Versive Seizure</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMPLEX MOTOR SEIZURE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automotor Seizure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hypermotor Seizure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gelastic Seizure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPECIAL SEIZURE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Atonic Seizure</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Negative Myoclonic Seizure</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actatic Seizure</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hypomotor Seizure</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hypnopomptic Seizure</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LATERALIZING SIGNS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ictal dystonia*</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automotor Seizure with preserved awareness</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Todd's Paralysis*</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Figure of 4*</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M2e sign*</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Postictal aphasia</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nose wiping *</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paradoxical lateralization (At end)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix B

Source Code in Ruby On Rails

B.1 Access Control Mechanisms of cRBAC

# Return true if the user has the given role in currently selected facility
def has_role?(role_name)
    # return false unless self.facility_selected?
    if role_name == "any"
        self.user_core_roles.find(:all).each do |tuple|
            return true
        end
    else
        self.user_core_roles.each do |tuple|
            return true if tuple.role and tuple.role.name == role_name
        end
    end
    return false
def any?
    has_role?('any')
end

# Return true if the user has the given role in given facility
def has_role_in_core?(role_name, core_id)
    if role_name == "any"
        self.user_core_roles.where(core_id: core_id).each do |ufr|
            return true
        end
    else
        self.user_core_roles.where(core_id: core_id).each do |ufr|
            return true if ufr.role and ufr.role.name == role_name
        end
    end
    return false
end

def has_role_with_global_privilege?(privilege)
    self.roles.each do |role|
        return true if (role.privilege_string.include? 'GLOBAL' ✓
                        and role.privilege_string.include? privilege)
    end
    false
end
#return true if user has given role

def has_role?(role)
    self.roles.each do |r|
        if r==role
            return true
        end
    end
    return false
end

# return true if user has given privilege in any core

def has_privilege?(privilege)
    self.user_core_roles.each do |ucr|
        if ucr.role.privileges.include?(privilege)
            return true
        end
    end
    return false
end

#

def has_privilege_in_core?(privilege, core_id) # In Parameter ← Facility
    return true if @privileges.has_key?(core_id) and @privileges[core_id].include?(privilege)
    # Return true if the user has a role in some facility that ←
    # has the global privilege
    return true if self.has_role_with_global_privilege?(privilege)
B.2 Implementation of Tider in Ontology guided Data Capture

```html
<tbody id='edit_epileptogenic_zone'>
  <% if diagnosis == 'Epileptic Paroxysmal Episode' %>  
  <tr>
    <td>  
      <span class='report-label'>Epileptogenic Zone</span>
      <div class="btn-group" style='float:right;'>
        <button type="button" class="btn btn-primary dropdown-toggle" data-toggle="dropdown">  
          Add <span class="caret"></span>
        </button>
        <ul class="dropdown-menu">
          <% EpileptogenicZone.where("parent_id is null").each do |zone| %>
            <li>  
              <%= get_dropdown(zone, "zone_#{zone.episodes_classification.id}", 'zone') %>
            </li>
          <% end %>
        </ul>
      </div>
    </td>
  </tr>
<% end %>  
</tbody>
```
<table>
<thead>
<tr>
<th>Epileptogenic Zone</th>
<th>Modifier</th>
<th>Has Modifier</th>
<th>Remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>#cz.modifier</td>
<td>#cz.epileptogenic_zone.name</td>
<td>vz.epileptogenic_zone.epileptogenic_zone.epileptogenic_zone.epileptogenic_zone.epileptogenic_zone.epileptogenic_zone</td>
<td></td>
</tr>
</tbody>
</table>
def get_dropdown(concept, container_id, tag_name):
    modifier = concept.has_modifier
    if modifier.blank?
        modifier = 0
    end
    if concept.children.empty?
        return "<li#{link_to_function(concept.name, 'addTag($({this}))';, container_id: container_id, tag_name: \rightarrow
tag_name, modifier: modifier)}"></li>"
    else
        result = ""
        concept.children.each do |child|
            result = result + get_dropdown(child, container_id, \rightarrow
tag_name)
        end
        return "<li class='dropdown-submenu'#{link_to_function(\rightarrow
            concept.name, 'addTag($(this));', container_id: \rightarrow
            container_id, tag_name: tag_name, modifier: modifier, \rightarrow
class: 'dropdown-toggle', 'data-toggle'='dropdown')} #{result}</ul></li>"
B.3 Matching algorithm for VISAGET

class TrialCriterion < ActiveRecord::Base
  attr_accessible :criterion_id, :max, :min, :trial_id, :value, :values, :group_position
  belongs_to :trial
  belongs_to :criterion
  has_many :group_criterions
  def get_categories
    categories = self.values.split(",").collect{|v| v.strip.downcase} if self.values
    categories ||= []
  end

  def categories
    categories = self.values.split(",") if self.values
    categories ||= []
  end

  def description
    c = self.criterion
    desc = c.description if c
    if c.boolean?
      return desc.gsub(’?’ , ”#{c.get_value(value)}”)
    elsif c.continuous?
      ...
    end
if self.min==c.minimum
    return desc.gsub('?', "less than #{max} #{c.unit}")
elself self.max == c.maximum
    return desc.gsub('?', "greater than #{min} #{c.unit}")
else
    return desc.gsub('?', "between #{min} and #{max} #{c.unit}")
end
e1s1f c.categorical?
    if self.get_categories.size > 1
        return desc.gsub('?', "one of #{get_categories.join(", ")")
    else
        return desc.gsub('?', "#{get_categories.join(", ")")
    end
e1s1f c.dynamic_categorical?
    if self.get_categories.size > 1
        return desc.gsub('?', "one of #{get_categories.join(", ")")
    else
        return desc.gsub('?', "#{get_categories.join(", ")")
    end
def boolean_value
    return self.value ? ["male", 'm', "1", "yes"] : ["female", "f", "0", "no"]
end
def get_value
    c = self.criterion

    if c.boolean?
        if c.name.downcase == 'gender'
            return self.value ? "male" : "female"
        else
            return self.value ? "yes" : "no"
        end
    elsif c.continuous?
        return "#{self.min}−#{self.max} #{c.unit}"
    elsif c.categorical?
        return "#{self.values}"
    elsif c.dynamic_categorical?
        return "#{self.values}"
    end
end

def precise_min
    if min.to_i == min
        return min.to_i
    else
        return min
    end
end

def precise_max
    if max.to_i == max
def match(patient)

c = self.criterion
# No patient data specified for this criterion, no way to ←
m Match
if c.patient_data.blank?
    return true
end
# puts "I am here"
value = patient.get_latest_data(c.patient_data)
# principal here is to avoid false negative
if value.blank? or value=='any'
    return true
end

if c.continuous?
    # puts "patient #{patient.id}, criterion #{criterion.id}"
    if c.unit.include? "limit"
        ref_range = patient.get_latest_data("#{c.patient_data.←
            split("-").first}._ref_range")
        ref_high = ref_range.strip.split("-").second
        ref_low = ref_range.strip.split("-").first
        ref_high = ref_high.to_f
ref_low = ref_low.to_f
end
begin
  value = value.strip.split.first.to_f if value.class==String
rescue
  # patient data is not correctly collected
  puts "patient data exception, treated as eligible favoring principle of avoiding negative false"
  puts c.patient_data
  puts value
  return true
end

if c.unit.include? "upper limit"
  calculated_value = value/ref_high
elsif c.unit.include? "lower limit"
  calculated_value = value/ref_low
else
  calculated_value = value
end

if calculated_value > self.max or calculated_value < self.min
  return false
else
  return true
end
end
if c.boolean?
    return self.boolean_value.include?(value.strip.downcase)
end

if c.categorical?

  value = value.strip.downcase

if c.name.downcase.include?("tnm")
  # if self.id == 1294
  #  p self.get_categories
  # end
  self.get_categories.each do |cat|
    if self.tnm_match(cat, value)
      return true
    end
  end
return false
elsif c.name.downcase == 'm'

  # matching result of m relies on value of patient ← metastasis
  # possible m values: 0, 1a, 1b
  # case 1: criteria only contains 0
  categories = self.get_categories
if value == 'x' or categories.select{|c| c=='x'}.present?
  return true
end

if categories.size ==1 and categories.first == '0'
  if value !='0'
    return false
  elsif patient.metastasis=='yes'
    return false
  else
    return true
  end
end

# case 2: criteria contains 1
if categories.select{|c| c.include? '1'}.present?
  if value=='0' and patient.metastasis=='yes'
    return true # because right now m value is unknown, ←
    # we count it to avoid false negative
  end

# if m is 1, 1a or 1b, then the matching is not ←
  relying on metastasis
if value.include? '1'
  result = categories.include? value
  return result
end
end

return true # if criteria are empty or not correctly configured
elsif c.name.downcase == 'n' or c.name.downcase == 't'
categories = self.get_categories

if value == 'x' or categories.select{|c| c=='x'}.present?
  return true
end

if not (self.get_categories.include? value)
  return false
else
  return true
end

elsif not(self.get_categories.include? value)
  return false
else
  return true
end

end

if c.dynamic_categorical?
  # Primary Diagnosis is a dynamic categorical criterion
value = value.strip.downcase

if c.name.downcase.include?("tnm")
    self.get_categories.each do |cat|
        if self.tnm_match(cat, value)
            return true
        end
    end
    return false
else
    if not self.get_categories.include? value
        return false
    else
        return true
    end
end
end
end

def tnm_match(v1, v2)

    vl_index_of_t = v1.index("t")
    vl_index_of_n = v1.index("n")
    vl_index_of_m = v1.index("m", vl_index_of_n)

    vl_value_t = v1[1..(vl_index_of_n-1)]
    vl_value_n = v1[(vl_index_of_n+1)..(vl_index_of_m-1)]
    vl_value_m = v1[(vl_index_of_m+1)..-1]
v2_index_of_t = v2.index("t")
v2_index_of_n = v2.index("n")
v2_index_of_m = v2.index("m", v2_index_of_n)

v2_value_t = v2[1..(v2_index_of_n-1)]
v2_value_n = v2[(v2_index_of_n+1)..(v2_index_of_m-1)]
v2_value_m = v2[(v2_index_of_m+1)..-1]

if(v1_value_t!='x' and v2_value_t!='x' and (v1_value_t!=← v2_value_t))
    return false
end

if(v1_value_n!='x' and v2_value_n!='x' and (v1_value_n!=← v2_value_n))
    return false
end

if(v1_value_m!='x' and v2_value_m!='x' and (v1_value_m!=← v2_value_m))
    return false
end

return true
B.4 Active Dashboard

B.4.1 Interface Development

<!-- Deceased or (Not Interested and has no surgery info) -->
<%= form_tag search_patients_path, method: :get, class: "form← search", id:"search-form" do %>
<%= hidden_field_tag 'sort', params[:sort] || 'last_name' %>
<%= hidden_field_tag 'direction', params[:direction] || 'asc' %>
<% do %>
<%== text_field_tag "search_field", @parameters[:search_field← ], placeholder: "Search patients by name or mrn...", class: "search-query span4", autocomplete: 'off' %>

<% off%== check_box_tag "deceased", 1, @parameters[:deceased].← present? %>
<%== check_box_tag "inactive", 1, @inactive % Show Inactive
<%== submit_tag "Go", class: "btn btn-info", id: "go-search← " %>
<%== submit_tag 'Export', class: "btn btn-info" %>
<%== link_to 'Create Patient', new_patient_path, class: 'btn ← btn-primary' %>
</div>

<% class='panel-padding panel-bordered panel-margin' id='cc_form' style='display:none;'>
<% text_field_tag "cc_date", Date.today.strftime("%Y-%m-%d") %>
<% link_to "Care Conference", '', class: "btn btn-primary btn-small", id: 'add_cc' %>
</div>

<div id='patients'>
<table class='table table-striped table-condensed' id='patient_index'>
<thead>
<tr class='pretty'>
<th><%= check_box_tag '', 0, false, id: 'select-all' %></th>
<th><%= sortable "last_name", 'Name' %></th>
</tr>
</thead>
<tbody>
<% Status.order('ordering').each do |status| %>
  <% if @parameters[:status] and @parameters[:status] =~ %>
    <% status_index = @parameters[:status]["#{status.id}" ] %>
    <% status_class = PatientsHelper::Status_Filter[status_index] %>
    <% end %>
  <% status_index ||= '0' %>
  <% status_class ||= 'status-box' %>
  <%= hidden_field_tag "status[#{status.id}]", '' %>
  <%= div class='%= status_class %'> onclick='changeStatusFilter(this, "status_<%= status.id %>")' %>
</tbody>
</table>
</div>
function sort_patient(column, direction){
    var current_column = $('.sort').val();
    if(current_column == column){
        var current_direction = $('.direction').val();
    }
}
if(current_direction == 'asc'){
    $('#direction').val('desc');
} else {
    $('#direction').val('asc');
}
} else {
    $('#sort').val(column);
    $('#direction').val('asc')
}
$('#search-form').submit();

function changeStatusFilter(element, id){
    if($(element).prop("class")=="status-box"){
        $(element).removeClass("status-box");
        $(element).addClass("status-box-warning");
        $('#'+id).val('1');
    } else if($(element).prop("class")=="status-box-warning"){
        $(element).removeClass("status-box-warning");
        $(element).addClass("status-box-success");
        $('#'+id).val('2');
    } else if($(element).prop("class")=="status-box-success"){
        $(element).removeClass("status-box-success");
        $(element).addClass("status-box-danger");
        $('#'+id).val('-1');
    }
else if($(element).prop("class") == "status-box-danger"){
    $(element).removeClass("status-box-danger");
    $(element).addClass("status-box");
    $('#'+id).val('0');
}
// searchPatient();
}

$('.status-box, .status-box-success, .status-box-danger, .status-box-warning').tooltip();

$('#select-all').bind('click', function(){
    // Jquery 1.6 +
    if($(this).is(':checked')){
        $('.select-patient').prop("checked", true);
        $('#cc_form').show();
    }else{
        $('.select-patient').prop("checked", false);
        $('#cc_form').hide();
    }
});

$('#cc_date').datepicker();

$('#add_cc').bind('click', function(){

var n = $( "select-patient: checked" ).length;
if (n == 0) {
    alert("No patient is selected");
} else {
    var patient_ids = [];
    $( ‘select-patient: checked’ ).each(function() {
        patient_ids.push($(this).val());
    });
    var cc_date = $(‘#cc_date’).val();
    $.ajax({
        url: "<%= batch_plan_care_conferences_path %>",
        data: { date: cc_date, patient_ids: patient_ids },
        type: "POST",
        dataType: "script"
    });
    return false;
});

</script>

B.4.2 The Query Algorithm

def search
    @parameters = params
    # patients_search are patients filtered by search field
    @search = params[ :search_field ]
@patients_search = (Patient.name_like(params[:search_field]) +
    Patient.mrn_like(params[:search_field])).uniq
# patients_status are patients filtered by status filters
@inactive = params[:inactive]
if @inactive.blank?
    @patients_search = @patients_search.select{|p| (p.deceased==false) and (p.surgeries.present? or (p.current_workflow and p.current_workflow.interest_in_dbs!='no'))}
end

params[:status].each do |key, val|
    if val == "-1"
        status = Status.find_by_id(key.to_i)
        if status==Status.find_by_short_name('I')
            @patients_search = @patients_search&.(PatientStatus.where(status_id: status.id, active: false).collect(&:patient))
        else
            @patients_search = @patients_search-(status.patients)
        end
    end
end

if val == "1"
    status = Status.find_by_id(key.to_i)
    @patients_search = @patients_search&(status.→ scheduled_patients)
if val == "2"
    status = Status.find_by_id(key.to_i)
    @patients_search &= (status ← completed_and_archived_patients)
end
end

@size = @patients_search.size

if params[:commit] == 'Export'
    @patients = Patient.where("id in (?)", @patients_search.←
            collect{|p| p.id}).order("#{params[:sort]} #{params[:←
            direction]}")
    @pdf = PatientExportPdf.new(@patients)
    send_data @pdf.render, filename: "export.pdf", type: "←
            application/pdf", disposition: 'inline'
else
    @patients = Patient.where("id in (?)", @patients_search.←
            collect{|p| p.id}).order("#{params[:sort]} #{params[:←
            direction]}")).paginate(page: params[:page], per_page:←
            20)
    respond_to do |format|
        format.html {
            render 'index'
        }
    flash[:notice] = "Search finished"
    format.js #search.js.erb
end
end
end
Bibliography


