

FACILITATING PATIENT AND ADMINISTRATOR ANALYSES OF ELECTRONIC
HEALTH RECORD ACCESSES

BY
ERIC LYNN DUFFY

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2013

Urbana, Illinois

Adviser:

Professor Carl A. Gunter

Abstract

The past two decades in the United States have ushered in an era of increasing ubiquity of digitized healthcare as the speed and sophistication of technology follows an ever-growing trend. Electronic health records (EHRs) are an integral part of the growing healthcare industry which offers ease of access and new functionality while simultaneously causing worries over their privacy and security. In an effort to address these concerns, much legislation has been enacted in order to tighten the oversight and requirements for accessing protected health information (PHI). Most recently, the Department of Health and Human Services has released rulemaking which requires providers utilizing EHRs to comply with patients' requests for logs of the accesses to their records.

In this work, we outline our system for complying with this regulation while easing the burden of compliance for providers and simultaneously providing patients with informative and satisfying information about why their accounts were accessed. We implement a system called the Multiview Audit Interface (MAI) which utilizes recent research in the data mining and anomaly detection communities to provide a unified interface for conveniently using these algorithms for patients and administrators. We then test this system on a de-identified access log from Northwestern Memorial Hospital containing months of audit data. We construct a framework for implementing these algorithms as modules, thereby recycling existing code, encouraging multi-faceted comprehensions of their results, and offering an easy-to-use interface that administrators and patients can use alike. We demonstrate the the power of three modules currently implemented and show how the extensibility of the framework can be harvested to develop modules in the future.

Table of Contents

- List of Tables v
- List of Figures vi
- Chapter 1 Introduction 1
- Chapter 2 EHR Overview 3
 - 2.1 Benefits and Risks 3
 - 2.1.1 Quality, Speed, and Convenience 3
 - 2.1.2 Coordination of Information 4
 - 2.1.3 Diagnostic Assistance and Error Prevention 4
 - 2.1.4 Patient Involvement 5
 - 2.1.5 Cost and Efficiency 5
 - 2.1.6 Roadblocks to Adoption 5
 - 2.1.7 Security Concerns 7
- Chapter 3 Regulations 8
 - 3.1 HIPAA 8
 - 3.1.1 Privacy 8
 - 3.1.2 Security 9
 - 3.2 HITECH 9
 - 3.2.1 Achieving Meaningful Use 10
 - 3.2.2 Subtitle D 10
 - 3.3 OCR NPRM for Accounting of Disclosures 11
 - 3.4 Access Reports 11
- Chapter 4 Design Requirements 14
 - 4.1 Cross-platform Compatibility 14
 - 4.2 Real-time Responsiveness 15
 - 4.3 Separation of Concerns 16
 - 4.4 Code Reuse 16
 - 4.5 Module API Design 17
 - 4.5.1 Architecture 17
 - 4.5.2 The Module Lifecycle 17
- Chapter 5 Modules 20
 - 5.1 Notes and Orders 20
 - 5.2 Position Explainer 20
 - 5.3 CADS 21
- Chapter 6 Explanations/Standardized Mapping 22
 - 6.1 Role and Access Coverage 22
 - 6.2 Potential Uses 24

Chapter 7 Implementation	26
7.1 Model View Controller Architecture	26
7.1.1 User Interface	26
7.1.2 Database	28
7.1.3 Business Logic	30
7.2 Module API Implementation	34
7.2.1 The Module Lifecycle	34
7.2.2 Module Java Interfaces	34
7.3 Module Implementations	36
7.3.1 Notes and Orders	36
7.3.2 Position Explainer	36
7.3.3 CADs	37
Chapter 8 Demonstration and Use Case	41
8.1 Patient View	41
8.2 Administrator View	42
8.3 Use Case Discussion	45
Chapter 9 Discussion	47
9.1 Affects on Healthcare Information Technology	47
9.2 Limitations and Future Work	48
References	49
Appendix A Database Schema	51
A.1 Schema Diagram	51
A.2 Table Descriptions	52

List of Tables

2.1	Survey results from EHR adopters [21]	5
3.1	A sample of a de-identified access log	12
3.2	A list of the nine most active positions within NMH	13
6.1	An example of NMH positions mapped to HPTCs	22
6.2	Uncovered Roles	23
6.3	ICD-9 code coverage of mapped positions	24
7.1	Statistics of the NMH access log data set	31
7.2	Capabilities and Interfaces	34
8.1	Summary of the diagnoses for patient 35347	41

List of Figures

4.1	Architecural Diagram of MAI	16
6.1	Uncovered Accesses	23
7.1	Example of the coloring XML returned from the <code>Coloring</code> servlet	27
7.2	Example of the description XML returned from the <code>Descriptions</code> servlet	28
7.3	SQL query to select all needed information about accesses for the given <code>patient_id</code>	33
7.4	A subset of the diagram relevant to the query in 7.3	33
7.5	Architecure Component Detail	33
7.6	Current Module API Java Interfaces Hierarchy	35
7.7	Query to retrieve Notes and Orders data	36
7.8	Registration XML for the Notes and Orders module.	37
7.9	Registration XML for the Position Explainer module.	38
7.10	SQL query issued to construct the entire access log for CADS	38
7.11	Registration XML for the CADS module.	40
8.1	The MAI welcome page	42
8.2	The PositionExplainer module at work	43
8.3	The MAI administrator display page	43
8.4	A lightly-colored user affected by the CADS module	44
8.5	A darkly-colored user affected by the CADS module	45
8.6	A combination of CADS, Notes and Orders, and Position Explainer modules	46
A.1	Entity Relationship Diagram	51

Chapter 1

Introduction

The primary focus of this paper will be on the security of electronic health records (EHRs) within the United States and the context of its legislative and regulatory environment. EHRs are a collection of digital information about an individual which can be accessed by various actors according to, or in violation of, over-arching security protocol.

The advent of EHRs within the last few decades has led to an unprecedented growth in the utilization of new technology in the domain of healthcare. EHRs have the potential to revolutionize the way that healthcare is conducted by reducing costs, preventing human errors, increasing the speed of communication, and advancing research.

In spite of this, the explosive growth of information technology in general has led to an influx of concerns regarding safety and privacy from the general population. There have been numerous examples of personally sensitive data being stolen, such as social security numbers, credit card data, and passwords to sites containing private information. Likewise, the healthcare information technology industry is no more immune to these concerns, and it is often an even greater target for parties interested in illegally acquiring personal information. EHRs often combine medical diagnoses, prescription information, credit card data, addresses, and much more.

To address these concerns, the United States congress has enacted a large body of legislation which regulates what information is deemed sensitive, who can legally view or modify such information, and what patients can do to protect and obtain their own information. Currently, patients in the United States have the right to request an accounting of all disclosures of their protected health information (PHI) as well as a record of all entities who access it. Such a right gives patients an abundance of information into the inner workings of healthcare provider organizations, potentially to the point of confusing the patient and to the dismay of organizational administrators.

In an effort to help healthcare organizations comply with regulatory requirements as well as to assist patients in understanding their own access records, we have constructed a system for harvesting the fruits of ongoing research in the machine learning and anomaly detection communities. This system, called MAI,

provides both patients and administrators with tools for interacting with analytical algorithms to yield constructive information about patients' access records. The system currently supports two functionalities: applying coloring rules to the access log elements which correspond to alert levels and providing descriptions of those elements.

The academic community has put forth a wealth of research designed so address the problem of enforcing patient privacy. Experience-based Access Management (EBAM) seeks to ensure that the continuously-evolving ideal access policies do not deviate far from their real-world counterparts [16] by analyzing the way that EHRs are actually accessed. The Explanation-based Auditing System (EBAS) evaluates the structure of the audit logs to attempt to determine an explanation for the accesses therein [9]. Other techniques use audit logs to analyze the access pattern of actors within an EHR system. These include the Community-based Anomaly Detection System (CADS) [7], the Patient-Flow Anomaly Detection System (PFADS) [30], and Specialized Network Anomaly Detection (SNAD) [8]. These techniques have been unified within the Extensible Medical Open Audit Toolkit (EMOAT) [1] to model and analyze EHR access logs. MAI is a natural extension to the EMOAT project to achieve the goal of increasing patient and healthcare personnel understanding of access logs.

The EHR vendor industry has also produced its share of solutions toward solving the problems outlined in this text. Cerner Corporation has released an audit tool known as P2Sentinel, designed to gather audit data and present the security personnel with reports and other interfaces for understanding the audit logs [3]. FairWarning is a company that has produced audit log analytical software to work with multiple EHR vendors. FairWarning's software analyzes the audit logs for suspicious patterns and alerts the security personnel when one is detected [2].

We will begin with a discussion of the present state of electronic health records in Chapter 2 followed by a discription of the legislative and regulatory history around EHRs in Chapter 3. Next we describe the design of our system in Chapter 4 before describing the modules currently implemented to work in the MAI framework in Chapter 5. We discuss how standardized role mappings can be utilized to improve patient and administrator understanding in Chapter 6. In Chapter 7 we cover the implementation details of the MAI system. Lastly, we give a demonstration and use case in Chapter 8 followed by a discussion of the project in Chapter 9.

Chapter 2

EHR Overview

An electronic health record (EHR) is a collection of digital data about a patient which is encapsulated in such a way as to be more-or-less equivalent to a traditional paper health record but one which is more conducive to electronic applications. EHRs can contain any data which is capable of taking a digital form and run the gamut from simple text-based nurse's notes to prescriptions, x-ray images, blood test results and more. EHRs are widely considered to be crucial to the modernization of the healthcare industry as they offer tremendous benefits over their paper-and-ink counterparts. In spite of their considerable benefits, EHRs have been met with numerous concerns regarding their cost of implementation and perceived risks to privacy.

2.1 Benefits and Risks

EHRs facilitate the ease of record-keeping, removing the need for large storage areas holding paper records, and eliminate the need for manually transferring records between rooms in an institution, institutions in a city or state, as well as between countries. Compatibility issues aside, EHRs have the potential to be accessible from any hospital, clinic, or pharmacy in any part of the world. EHRs are also conducive to data analysis which can help prevent missed life-saving diagnoses, erroneous prescriptions as well as deadly medication reactions. EHRs have also been utilized to provide a wealth of research material, in de-identified form, which is invaluable to researchers across the world.

2.1.1 Quality, Speed, and Convenience

Possibly the single most important aspect of the information technology revolution is the ability to program a computer to do mundane tasks that would require a human hours, days, or longer to complete. Likewise, EHRs have the potential to reduce the amount of paper-pushing and routine procedures often associated with the medical industry. Rather than spending time looking for a paper health record which could be misplaced, missing, or stolen and is incapable of duplication, an EHR can be instantly accessible from any

location in the world with 100% reliability.

EHRs also eliminate the need for redundantly completing information. Once data is entered into an EHR, it remains there until explicitly removed or updated. E-prescriptions can be transferred from clinic to pharmacy in milliseconds, thereby eliminating the need for handling forgery- and error-prone prescription pads. Data analysis can be performed on a patient's health data to render a diagnosis which could take a human being hours of poring over medical references to discover, regardless of his or her experience. EHRs also provide built-in safeguards to alert medical staff of dangerous events such as drug interactions [18].

2.1.2 Coordination of Information

It is not unusual for a patient to be seen by a multitude of personnel, even for the most routine checkups. Patient health data must be shared and coordinated efficiently between administrative staff, nurses, doctors, radiologists and specialists to only name a few. Each of these individuals contribute to the care of a patient in a limited way which must all sum to an effective treatment. To make healthcare more efficient, this fragmented interaction with a patient's care needs to be mitigated.

EHRs help ensure that each separate provider has as complete a picture of their patients as possible. They eliminate the need for redundancy and the risk of data loss when transitioning between different environments within an organization or between organizations. This greater availability of patient health information also means that data is available when it is critically needed, such as when a patient is in an emergency room and cannot provide his or her information [19].

In order to prevent the abuse of this abundance of information, much research has been conducted to determine the best ways to protect patient data from unauthorized persons while simultaneously making it available to those who have a legitimate need for it.

2.1.3 Diagnostic Assistance and Error Prevention

Maintaining a centralized point of reference for all of a patient's medical information yields a great potential for exploiting this wealth of data to assist medical professionals in making decisions. Computers are becoming increasingly powerful and can find patterns in data that a human being would struggle to identify even if he or she is searching for it. EHRs can aggregate a list of patient allergies which can be cross-referenced with any drugs about to be administered to a patient, thereby averting a deadly reaction if the doctor is otherwise unaware of the allergy. An EHR is aware of which medications a patient has been prescribed even if the patient is uncomfortable making their doctor aware of his or her medical history due to its sensitive nature. Computational power can also be leveraged to identify the best way to code a diagnosis or procedure since

Statement	Percentage Agreeing
Sending Rx electronically saves time	82
Saves on managing & storing paper costs	75
Overall, their practice functions more efficiently	79
An asset when recruiting physicians	68
Receive lab results faster	75
Produces financial benefits for the practice	67
Enhances data confidentiality	70

Table 2.1: Survey results from EHR adopters [21]

it is virtually impossible for a doctor to be aware of all available codes [20].

2.1.4 Patient Involvement

Since the development of the Internet, the world has become a much more connected and collaborative place. Now the technology is available for a patient to easily interact with his or her care providers at any time and from any place. EHRs encourage this kind of distributed collaboration by making health data readily accessible for both the patient and the provider. A patient can now update his healthcare provider with new information whenever it is most convenient for the patient and this results in more timely treatment of new maladies.

The development of the Personal Health Record (PHR) has surfaced in recent years. These records are chiefly maintained by the patient itself, and allow the patient greater accountability for his or her own health. Patients are now able to fill in data in a guided fashion which can encourage preventative treatment and early detection of oncoming disease.

2.1.5 Cost and Efficiency

According to a recent national survey of meaningful-use ready doctors, EHRs have a palpable impact on the efficiency of healthcare organizations. In each of the statements shown in Table 2.1, more than two-thirds of respondents agreed that efficiency had increased since the adoption of an EHR. These results suggest that there is a considerable gain in efficiency within hospitals across the country due to implementing EHRs.

2.1.6 Roadblocks to Adoption

EHR systems, by their nature, are more conducive to being adopted by large organizations rather than smaller ones. Most of the reasons for this are outlined below: EHRs require a massive commitment from an organization in terms of money, time, talent, and legal resources. In order for widespread EHR adoption to become a reality, the following issues must be addressed. [15]

Financial Burdens

The financial burden of EHR implementations is probably the single largest barrier to their widespread adoption. Not only are EHRs associated with a high cost of implementation, but there is a good deal of uncertainty about their true return for the investment. Implementing an EHR consists of a number of complex phases including conversion of paper information to electronic data, implementing a system that works reliably within each organization, configuration to work with existing systems, training the staff to use it, and maintaining the EHR as the organization evolves over time.

Technological Maturity Concerns

New technologies are innately unstable, and EHRs are no exception. It is not uncommon for the more conservative consumer to wait some time before he or she begins using a new technology, allowing the early adopters to handle all of the bugs and difficulties in early versions of a system. It is no wonder, then, that some healthcare providers would prefer to not adopt a new technology especially when that technology requires a vast commitment.

Some trained healthcare professionals do not use computers much at home and therefore have little training that would encourage EHR adoption. Furthermore, EHR systems require large amounts of data for even mid-level providers and can be exceedingly complex. This can lead to a scenario in which applications running on top of the EHR system are sluggish to respond, are unreliable in behavior, or may even experience data corruption. The Certification Commission for Health Information Technology (CCHIT) has sought to remedy these concerns by giving criteria for evaluating EHR systems.

Complying with Regulations

EHRs, and healthcare in general, are associated with a large and complex body of legislation, much of which is covered in Chapter 3. While existing legislations take great strides to encourage the adoption of EHR adoption, there is still a cost-benefit analysis that a provider must consider before implementing an EHR system. There is currently no legal requirement for adopting electronic health records, but legislation encourages it by providing financial incentives.

A provider must consider several things when determining if an EHR system is worth implementing.

- How well will one be able to comply with new regulations? Much of the United States healthcare system reform has introduced new liabilities for providers implementing EHRs, such as offering the ability for a patient to acquire an access report.

- Who will be affected by new legal requirements after adoption? New avenues of achieving results also enlarge the amount of a provider which can be held accountable. E-prescribing, for instance, makes ordering prescriptions much easier but also potentially increases the number of physicians making legally questionable prescriptions.
- How will regulations about disclosures of PHI affect one's practice? The new ease of information dispersal accompanying EHRs increases the ability for data to be disclosed. A provider may be concerned that this encourages PHI being given to inappropriate parties.

2.1.7 Security Concerns

Given all of the potential costs and benefits of adopting an EHR system, it is paramount that the effort of making EHRs a reality not go to waste by allowing unrestrained access to patients' sensitive information. A centralized data store for all patients within a country has typically been poorly received in the United States. When the Health Insurance Portability and Accountability Act (HIPAA, see Section 3.1) of 1996 was passed, a national patient identifier (NPI) was required by law, but two years later federal funds were prohibited from being spent on developing the NPI following public outcry [6].

As a result, all of the patient information within the United States is distributed across multitudes of non-uniform systems which are provided by several vendors. In addition, it is not uncommon for patient health data to be disclosed to the business associates of healthcare organizations, thereby leading to an even larger distribution of health data. It is then understandable that so much effort would be put into place to protect patient health data such as the regulations that we will discuss in Chapter 3.

Chapter 3

Regulations

An abundance of legislation and regulation has been introduced into the United States legal system in order to address the concerns outlined in Chapter 2. In this chapter we will present the two core pieces of healthcare IT legislation, the Health Insurance Portability and Accountability Act and the Health Information Technology for Economic and Clinical Health Act, and how they present the need for the research in this text.

3.1 HIPAA

Healthcare information technology has a long history of oversight in the United States, but much of its modern roots can be traced back to the passage of the Health Insurance Portability and Accountability Act of 1996 (HIPAA). Prior to the enacting of this law, health insurance entities, now called “covered entities” had disparate means for conducting administrative business such as submitting claims. Furthermore, there were no uniform methods or standardizations for exchanging electronic data between healthcare entities nor were there modern guarantees about the security and privacy of electronic data.[5] To address these concerns, the United States congress passed HIPAA and, in doing so, included many provisions for protecting the privacy and security of protected health information.

3.1.1 Privacy

Compliance for the HIPAA Privacy Rule was required as of April 2003 [24]. The Privacy Rule regulations are intended to prevent the abuse of PHI and ensure that its use is always appropriate for medical care. Prior to HIPAA, most privacy laws applied to personal information such as financial information but failed to extend to medical data. In addition to insufficient existing laws, the healthcare privacy law was plagued with dozens of inconsistent state laws, mostly applying to sensitive diagnoses.[29].

The Privacy Rule uniformly addresses all of these problems by defining how protected health information may be legally disclosed by covered entities and their business associates. Essentially, any covered entity may disclose PHI without the patient expressing his or her authorization if the disclosure is for the purposes

of treatment, payment, or health care operations.

The nature of the PHI itself is defined by its individual identifiability; that is, if it is identified with an individual. This type of data includes all possible forms such as written, oral and electronic. De-identified health information has no restricted disclosure and is often used for research purposes.

3.1.2 Security

The Security Rule is a counterpart to the Privacy Rule which applies specifically to electronic health data. It defines a set of security standards required to be implemented to protect PHI. These standards were designed to follow best practices and increase patients' confidence that their PHI would not be improperly stored, discarded or otherwise leaked. Simultaneously, the Security Rule was written in such a way so as not to stifle innovation or prevent the full utilization of the benefits of EHRs

The first methods described in the Security Rule for ensuring the security of PHI are those of administrative safeguards. The management of a covered entity is required by law to "reduce risks and vulnerabilities to a reasonable and appropriate level." Information access should be governed by a role-based access control policy and each person should have minimal privileges possible according to his or her role.

Additionally, covered entities are required to fulfill physical security requirements. Not only must facilities be locked and managed appropriately, but safety measures must be taken moving or disposing of physical media. For instance, HIPAA requires that the hard disks from terminals containing PHI be wiped so that the data is irrecoverable.

Finally, technical safeguards must also be put into place to comply with the Security Rule. As previously mentioned, access controls must be put into place to prevent granting overly-generous privileges to employees of a covered entity. All PHI must be protected with integrity-checking mechanisms that prevent the alteration or destruction of data by unauthorized persons. When transferring data between entities, data must be protected by encryption to prevent eavesdropping or alteration. Covered entities are also required to implement audit controls. These mechanisms record the accesses made to PHI which can later be reviewed by security personnel or automated anomaly detection software.

3.2 HITECH

The wide-spread adoption of EHRs came about following the enactment of the Health Information Technology for Economic and Clinical Health Act (HITECH) of 2009 which was included as a title under the American Recovery and Reinvestment Act, providing economic stimulus spending. This act greatly incentivized the

“meaningful use” of EHRs, meaning that providers must implement and use EHRs in a way that improves the quality of care in order to take advantage of the incentives. [10] Beginning in 2015, providers who do not implement a certified EHR will be penalized a percentage of Medicare payments.

3.2.1 Achieving Meaningful Use

The Centers for Medicare and Medicaid Services (CMS) have divided the process of achieving meaningful use of EHRs into two broad steps, termed Stage 1 and Stage 2 [11]. Stage 1 consists of 15 mandatory requirements, called *Core Requirements* as well as 10 other *Menu Requirements* of which half must be met.

The Core Requirements largely consist of automated statistical data accumulation and reporting as well as direct electronic equivalents to existing paper record functions. The later category requires functionality such as maintaining electronic medications and allergy lists, issuing electronic prescriptions and providing the patient with an electronic copy of their PHI upon request. There are also a few more advanced functions which are currently technically tractable, including drug-drug and drug-allergy reaction checks and clinical decision support.

Stage 2 meaningful use has a similar structure, with 17 Core Objectives and 6 Menu Objectives, of which 3 must be met [14]. Most of the Core Objectives of Stage 2 are duplicates of those in Stage 1 but are accompanied by more stringent measurements, meaning that the use of EHRs and their associated applications must be more widespread [13]. Most of the Menu Objectives require the gathering of statistical information such as family health history or cancer cases and sometimes require submitting them to a specialized registry.

By incentivizing the widespread adoption of the meaningful use of electronic health records, the United States government is prioritizing the gathering of personal health data in large amounts and using those data in novel and varied ways. Rather than inhibiting the march of progress, the United States congress dedicated a portion of HITECH to the preservation of patient privacy and bolstering the provisions of HIPAA.

3.2.2 Subtitle D

HITECH Subtitle D was constructed to modify existing laws regarding the privacy and security of protected health information, largely through means of mandatory information disclosures by covered entities, performing audits and strengthening penalties for violators. Under Subtitle D, upon the discovery of a breach of PHI, a covered entity or the business associate of covered entity is required to notify not only the individuals affected but also the prominent media outlets[17]. Furthermore, the HHS Interim Final Rule for Subtitle D modifies the Social Security Act by establishing a tiered set of penalties based on the nature of the violations

and by eliminating some exemptions for covered entities which are unaware of violations as well as those who correct a violation within a timely manner [23].

3.3 OCR NPRM for Accounting of Disclosures

As part of the evolving regulatory landscape of the HITECH Act, the Office of Civil Rights (OCR) within the Department of Health and Human Services (HHS) published a notice of proposed rulemaking (NPRM) in May 2011 [22] which announced proposed rules for enforcing the legal document. Among numerous other changes, HHS stated their intention to extend the existing provision allowing patients to acquire an accounting of the disclosures of their health information. Prior to the rulemaking, exemptions had been included for accounting of PHI disclosures made for the purposes of treatment, payment and health care operations. The NPRM proposed, in contrast, to eliminate this exemption for covered entities utilizing electronic health records, believing that the overhead for implementing such functionality would be quite limited. Furthermore, HHS announced that patients would also be able to receive an *access report* which, in contrast to an accounting of disclosures, would include a summary of all individuals accessing their information, regardless of whether the information was disclosed to an external entity.

This new regulation simultaneously promises to give a patient great freedom over his or her own information in an electronic medical record by allowing the patient to know precisely who accesses his information and when. Unfortunately, this also presents a considerable burden on healthcare providers who must provide these reports as well as respond to any criticisms raised by the patients who receive them. Because the knowledge of patients and administrators alike is quite limited in the context of such a vast system as a modern healthcare provider, there needs to be a way to automate as much of the implementation of the NPRM as possible.

3.4 Access Reports

The NPRM proposed the following data to be present in the access report for each access to a patient's PHI:

1. The date of access.
2. The time of access.
3. The name of the person, if available, or entity accessing the information.
4. A description of the information accessed, if available.

5. A brief description of the action taken, if available.

According to the NPRM, HHS “expect[s] that all access logs include this information, so [they] believe it should be readily available for inclusion in access reports without substantial burden to covered entities and business associates.” HHS further does not require the presence of a reason of access because in doing so “the burden on covered entities and business associates in identifying the purpose of each access to electronic designated record set information significantly outweighs the benefit to individuals of learning of such information.” However, it seems likely that a patient, upon receiving a copy of his or her access report, could become concerned with the nature of the accesses, particularly if their record is accessed by individuals with unfamiliar role titles whom the patient has never encountered.

Table 3.1 contains a sample of an actual de-identified access log for a single patient extracted from Northwest Memorial Hospital in Chicago. In a real-world situation, the users would be identified by their real name. In the Northwestern Memorial Hospital access log, each user is assigned a position, or role, and the automatically-generated access reason is often similar to the role description. One can see that if a user has a highly-specialized role, the natural access reason available will not be clear to the patient.

Date	Time	Name	Reason
11/23/2010	6:33	User A	OR RN SC- Primary
11/23/2010	6:56	User B	Resident- Inpatient Primary Service
11/23/2010	7:17	User C	Anesthesiologist
11/23/2010	7:22	User D	OR RN SC- Primary
11/23/2010	7:31	User D	CRNA
11/23/2010	7:32	User E	OR RN SC- Primary
11/23/2010	8:44	User F	Auditing/Quality
11/23/2010	8:48	User D	CRNA
11/23/2010	8:52	User D	CRNA
11/23/2010	10:42	User G	OR RN SC- Primary
11/23/2010	10:46	User H	Med Student - Outpatient/ED/Procedures
11/23/2010	10:48	User B	Resident- Inpatient Primary Service
11/23/2010	14:32	User I	Primary Staff Nurse
11/23/2010	16:57	User F	Auditing/Quality
11/29/2010	13:44	User J	Billing
11/29/2010	20:39	User J	Billing
11/30/2010	12:21	User K	Physician Office

Table 3.1: A sample of a de-identified access log

Table 3.2 contains a list of the nine most prevalent positions within the Northwestern Memorial Hospital access logs. The positions in this table are ordered by the percentage of encounters accessed by that position. Typically an encounter is defined as the entire span of time that a patient is at a hospital, so each encounter can be thought of as a collection of accesses to an EHR. It is unlikely that a patient checking his or her access record will understand the purpose of any of these positions and it is possible that even a hospital

administrator would not know the meaning of some of them.

Position	% of Encounters
HIM	93.70%
UR/QA 1	52.36%
HIM - Specialist	16.87%
Reg/Sched Super-User	16.67%
NMPG MD - CPOE	5.24%
NMPG MA	4.25%
NMH Operations	3.79%
UR/QA 2	1.69%
Any of the above	95.69%

Table 3.2: A list of the nine most active positions within NMH

We believe that current technology is sufficiently advanced to address these concerns by assisting both the patients and administrators in learning about the nature of accesses. Statistical analysis and machine learning algorithms are capable of detecting patterns within vast amounts of data which can be leveraged to either explain legitimate accesses or to detect anomalous accesses. Using these techniques, patients can be given information about the accesses to their EHR in order to alleviate concerns over who is accessing their information. Meanwhile, administrators can utilize an interface to understand accesses in the event that a patient is still concerned with an access. He or she can then use that information to act accordingly.

Before we proceed in implementing a system for generating contextualized and descriptive access reports, we must first analyze the requirements for implementing such a system. It must be capable of responding to a varied set of users who are widely distributed geographically, responding in a satisfying manner, and flexible enough to envelop future developments.

Chapter 4

Design Requirements

In this chapter we outline the design requirements for the MAI system. This system must simultaneously satisfy an array of players who will use it:

- *Patients* must be able to access the system from any location and request an access report in a timely manner.
- *Security Administrators* must be able to use the system to gather evidence for or against an investigation into a HIPAA violation or simply the questions of a concerned patient.
- *EHR Administrators* must be able to easily configure MAI to work with their EHR implementation regardless of the technologies they use.
- *Researchers and Developers* must have a well-defined yet flexible framework for implementing analytical components into the system. This framework must also be accommodating to new and evolving technologies.

4.1 Cross-platform Compatibility

Because of the vast array of technologies employed in an EHR environment, it is important to be mindful when designing MAI of the need for cross-platform compatibility. It is impossible to rely upon patients and administrators to predominantly use a single operating system or have the necessary set of libraries to use MAI. With the emergence of smartphones and tablets it is increasingly apparent that supporting one or even two operating systems would prevent a large swath of the potential users from utilizing MAI.

For these reasons, we have decided to implement the front-end user interface as a web application. Virtually no consumer-grade electronics with internet access lack a web browser, so the full user base can take advantage of MAI with existing hardware.

In addition to having a widely-supported user interface, it is necessary to implement the business logic of MAI in a way such that it could be readily ported to virtually any EHR vendor. Our basic assumption is

that the audit logs of any EHR can be gathered in a tabular format such as a Structured Query Language (SQL) database and retrieved through simple SQL select statements. This assumption is reflected in the language of the Department of Health and Human Services's Notice of Proposed Rulemaking regarding HIPAA's privacy policy.

The MAI framework was therefore decided to be implemented on top of an Apache Tomcat Java container. The Java Enterprise Edition runtime has implementations for nearly all operating systems and architectures used in server environments. Each anomaly detection module used by MAI must then implement a Java interface to the tomcat server and respect the module lifecycle contract outlined in Chapter 7. Once the module's Java interface has been defined, the module is free to follow any code path it requires which includes querying for access log data, processing the data, and writing intermediate results to disk. At this point it is up to the module designer to uphold cross-platform compatibility or to disregard it entirely as he or she sees fit.

4.2 Real-time Responsiveness

The performance of each individual module must have the appearance of real-time responsiveness to the end user so that when he or she logs into MAI via the web interface, all analysis on the patient's records is instantaneously available. Unfortunately, many of the intended modules for use in MAI require a massive amount of calculations which can require hours or even days on powerful servers, depending on the size of the access log in question. Furthermore, many modules need some contextual information about the day-to-day activities within the hospital, how social networks develop over time, and how roles typically behave. Therefore most modules require more data than simply a single patient's access log and often the entire access log must be provided for stable results.

To address this issue, we have divided the module lifecycle into multiple phases outlined in Chapter 7 which includes a preprocessing stage. In this stage, a module is expected to perform computations on as much of the available access log data as it requires and then store the results so that they can be acquired later on by the end user. The module will then periodically be updated with incoming access log data and asked to renew its preprocessing.

A major drawback to this blocked approach is clear: if a patient's record has been accessed more recently than the latest module update, then that segment of the audit log will have no associated anomaly information. In a real world implementation, we would suggest alerting the administrator when he or she queries for anomaly data on a patient who has unprocessed accesses so that they don't mistakenly think that the

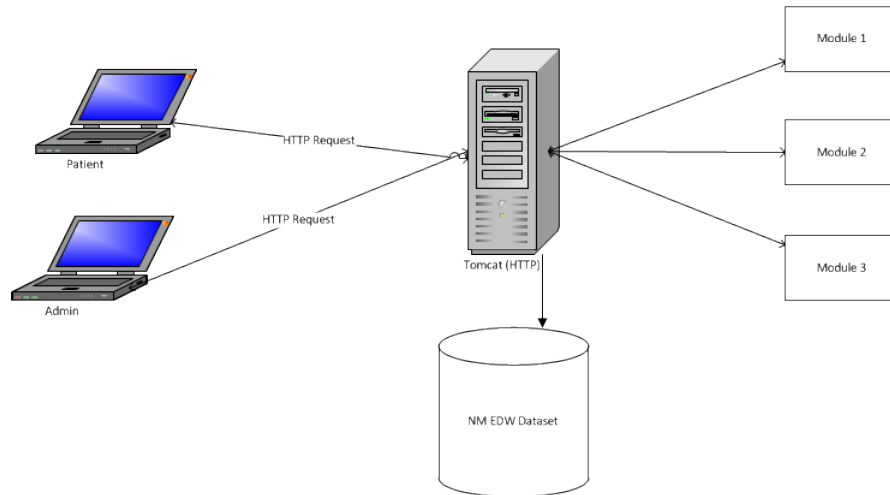


Figure 4.1: Architectural Diagram of MAI

modules have tried and failed to detect an anomaly.

4.3 Separation of Concerns

Broadly, the MAI architecture can be seen as a model-view-controller architecture. The user, a patient or administrator, logs into a webpage which displays access logs and information about the accesses, roles, and users. This component is referred to as the *front-end*. The core business logic is handled by a Java Enterprise Edition (Java EE) application running on a Apache Tomcat web server, collectively called the *middleware*. The middleware supports a plug-and-play interface for the analytical modules which must simply implement the necessary Java interface and be registered with the middleware. All of the data to be processed is contained within a normalized SQL database running on the server, or potentially elsewhere, and is referred to as the *back-end*. The high-level architectural diagram can be seen in Figure 4.1.

Because of the need for interchangeable analytical modules, we have designed and developed an application programming interface for the modules, called the *Module API* which is outlined in 4.5.

4.4 Code Reuse

MAI was designed to impose as few constraints on the module developer as possible while still treating each module as an abstract component. Each module must implement a Java class which is used to interface with the MAI middleware, but the modules should not be constrained to being programmed entirely in Java. Much of the machine learning and statistical analysis software used in these respective communities are

written in other languages, using other environments. In order to reuse code from Matlab or R, the module programmer is able to implement a Java wrapper class to interface with the middleware, and then it is free to use local storage, the network, call external processes, or any other programming construct it desires.

4.5 Module API Design

The purpose of the Module API is to allow distinct, interchangeable anomaly detection algorithms to operate in a uniform way on access log data from an Electronic Health Record (EHR). The current focus for this project is to allow a module to register with a web framework, extract all necessary data for preprocessing, and, upon requests in real-time, provide a measure of the degree to which an element in the EHR is anomalous. This must be done in a way that is independent of the algorithm's implementation, so that previously developed algorithms can be leveraged and so that the author of the algorithm is not constrained to any implementation decisions.

4.5.1 Architecture

The architecture of MAI is a three-tiered architecture consisting at a high level of the user's front-end applications, seen through a web browser, a middleware server hosting a Tomcat Java Servlet container, and a database containing the EHR access logs.

The middleware is responsible not only for responding to HTTP requests from the clients but also for coordinating the Module Lifecycle (Section 4.5.2). The middleware must allow module registration and removal, facilitate organization among modules, supply modules with the EHR access log data in an usable format, and requesting anomaly measurements for EHR components from the modules.

4.5.2 The Module Lifecycle

Here we will outline the distinct phases in a module's lifecycle in relation to its interaction within the web framework. The phases are, in chronological ordering, module registration, preprocessing, real-time queries (RTQ) and uninstallation. Module registration (Section 4.5.2) enables the MAI middleware to understand the functionality of a module and know how to shift the flow of execution to it. Preprocessing (Section 4.5.2) tells the module to begin gathering data, running analysis on it, and storing the results as needed. RTQ (Section 4.5.2) begins once preprocessing is complete and enables the end user to issue data queries which complete in real time. Lastly, uninstallation (Section 4.5.2) is called to let the module clean up any and all intermediate storage that it has used.

Module Registration

Each module must make the middleware aware of its existence and supply enough information for the middleware to know what information to provide and how to understand the module's output. This process is called Module Registration (MR) and is implemented by defining a well-formatted Extensible Markup Language (XML) [28] file containing the meta-data required by the middleware.

The requirements for this stage are:

1. Provide the name of the module.
2. Provide the class name and location of the Java interface for the module.
3. Give the preferred format for data needed for preprocessing, if any.
4. Define the parameters required for the algorithm to yield a result at runtime, i.e., does the module require a patient ID, user ID, timestamp, etc.
5. Define the return type for real-time queries.

Optional parameters are:

1. Provide extra information for more efficient processing so that, for example, multiple queries can be combined into a single query.
2. Provide a time frame for how often to update the module with new data.
3. Tell the server that it is or is not okay to cache results between data updates.
4. Provide a true or false value for whether the server should pre-fetch queries from the module before the user has selected the module. This reduces the latency seen by the user but may not be desirable if the module requires significant resources.

Once this information is defined in an XML document, the module can be registered and the middleware will update or append the master XML file which contains the behavior for each module as well as the protocol for interoperating the modules.

Preprocessing

Since many of the anomaly detection algorithms we will be using require a significant amount of training time, it is necessary to include preprocessing time in the MAI protocol. Preprocessing can begin at any time the middleware desires, so long as it has audit log data ready for consumption by the modules, by

calling the `preprocess()` method exposed by the class referenced in the module's registration XML file. The middleware will supply data in the format requested in the same XML file. The module must also implement a `preprocessingComplete()` method which will return true when the module has finished this stage.

Real-time Queries

Once the module returns true from the `preprocessingComplete()` method, the middleware will enable users to select the module via the front-end web application. The middleware will supply the module with the data defined in the registration XML file, as well as an API for making queries on the database. The module will return data in the format defined. The middleware will then construct the webpage for the client appropriately.

Uninstallation

If a module needs to be removed, it should provide a `cleanup()` method to destroy all of the persistent data it has created.

Chapter 5

Modules

In this chapter we outline the modules which have been thus far implemented in the MAI system. Each module has its own set of required parameters as well as its own information that it yields. Every module must be handled in a uniform manner by the MAI system, so we discuss how the architecture outlined in Chapter 4 addresses this. The implementation details for each module can be found in Section 7.3

5.1 Notes and Orders

The Notes and Orders module looks at the current state of the access log and looks for accesses for which there was a change to a note or an order. The core reasoning for this module is that if an EHR user, such as a nurse, examines a patient's record, they are unlikely to make a note or an order if the access is illegitimate in nature. When a user makes such a modification, the notes and orders are typically reviewed by at least one other member of the staff, therefore bringing attention to the access.

The Notes and Orders module requires no preprocessing or uninstallation time, but does need a database connection during the RTQ phase. In a strict sense, Notes and Orders only requires an `access_id` as input, but for efficiency purposes it should be given the `patient_id` as well so that batch processing can occur.

5.2 Position Explainer

In order to give both patients and administrators some textual information about why a user accesses a patient's record, we have developed the Position Explainer module. This module builds on the principles which will be discussed in Chapter 6 by calculating the likelihood of a user accessing a patient's record based on the patient's diagnosis information.

Each patient has zero or more ICD-9 diagnoses associated with them and patients with similar diagnoses tend to follow similar workflows. For example, when a patient is admitted to a hospital with a broken leg he or she will follow a routine procedure such as check in at reception, have a physical examination, go through radiology for an x-ray, have a cast placed on the leg, and discharge. It is highly unlikely that a patient with

a broken leg will need an abdominal ultrasound or a psychiatric screening, such as are common for expecting mothers and mental health patients, respectively. This principle means that many diagnoses have a high likelihood of being accessed by a particular user if that user has one of the common roles associated with the diagnosis.

The Position Explainer module requests a `patient_id` and `user_id` as input and returns a description if that user's position has a sufficiently high likelihood of accessing the patient based on his or her diagnoses. For each diagnosis belonging to the patient, the module determines the likelihood a role accessing a patient with that diagnosis and repeats this for each role belonging to a user in the patient's access log. If the likelihood is higher than a tunable threshold, the user is given a message stating that the user's position is highly associated with his or her diagnosis.

5.3 CADS

The community-based anomaly detection system (CADS) was developed to detect anomalous actors within a collaborative information system [7]. CADS works by building a social network that connects users who access the same patient record within an EHR. Once the social network has been constructed, CADS performs principal component analysis on the network to extract the principal communities within the network. After this decomposition, each user can be compared to his or her nearest neighbors via the k -nearest neighbors algorithm which then gives a deviation for each user from his or her community.

CADS requires considerable preprocessing time compared with the other two modules as it performs a number of complex calculations on a large data set. CADS asks for a `patient_id` during its session initialization phase so that it can quickly return its results without needing to repeatedly look up results for each user when only a small number of users are anomalous. It returns both a double as an anomaly score as well as a description yielding information about why the user was detected as anomalous.

Chapter 6

Explanations/Standardized Mapping

Since the position names within the NMH data set can be unclear, ambiguous, and difficult to understand, we hope to be able to soon be able to map these positions to a widely standardized set of values that have a clearer meaning. One way of achieving this is to use the Healthcare Provider Taxonomy Code (HPTC) Set [12], which is a hierarchical set of standardized position codes used in the United States.

6.1 Role and Access Coverage

We have acquired a mapping between the position names in the NMH database and the codes in the HPTC set, courtesy of Dan Cushman, which we hope to leverage for this purpose. An example of the mapping is shown in Table 6.1.

Unfortunately, this mapping is not complete. Of the 159 positions that we have in our data set, only 66 (41.5%) are mapped to HPTCs. We will refer to those positions with a HPTC code as “covered” positions or “covered” roles. Luckily, most of the more common positions do have mappings, as 4,984,334 out of the 6,075,530 (82.0%) accesses in database are made by a user with a position that is mapped to an HPTC. In fact, 44% of the accesses made by uncovered roles are due to a single role, “Patient Care Assistive Staff”. Nearly all (90%) of these uncovered accesses are attributed to 14 of the 93 (15.0%) uncovered roles, which are shown in Table 6.2. The distribution of the uncovered roles can be seen in Figure 6.1. Since the graph demonstrates a rather sharp decline in the coverage, we may be able to utilize the mapping that we have to discover a clearer structure from the positions in our data set.

NMH Position	HPTC	HPTC Description
Chaplain	374K00000X	Chaplain
Dietary 1	133V00000X	Dietitian, Registered
ED Patient Care Staff Nurse	163WE0003X	Emergency
NMPG APRN	364S00000X	Clinical Nurse Specialist

Table 6.1: An example of NMH positions mapped to HPTCs

Position	% of Uncovered Accesses
Patient Care Assistive Staff	44.3
Unit Secretary 1	11.6
UR/QA 1	11.2
NMH Physician Office - CPOE	5.7
HIM	5.4
Health Information Audit-Consultant	2.3
NMPG OB PEDS MD - CPOE	1.8
NMH Resident/Fellow ID Clinic-CPOE	1.5
ED Assistant	1.1
ED Unit Secretary	1.1
ED NMH Physician-CPOE	1.1
Unit Secretary 2	1.0
Reg/Sched Super-User	1.0
ED Coder	0.9

Table 6.2: **Uncovered Roles:** The roles that together account for 90% of the uncovered accesses.

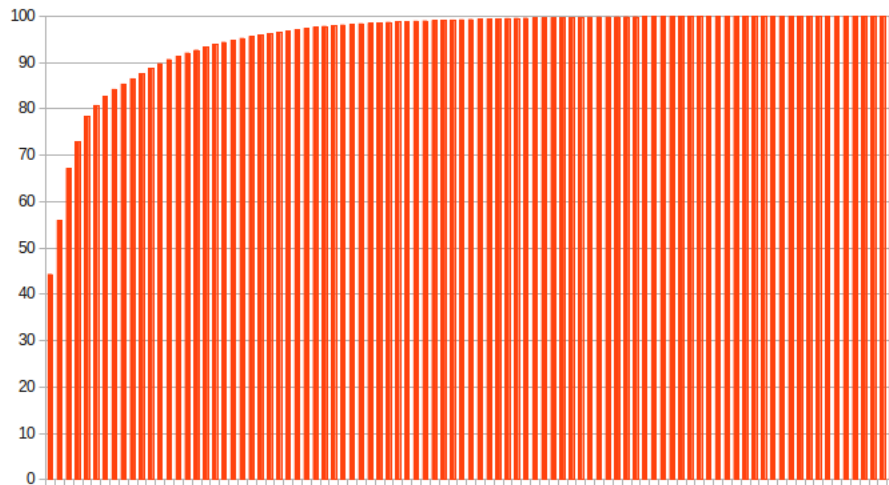


Figure 6.1: **Uncovered Accesses:** The percentage of accesses left uncovered by each role, sorted by decreasing coverage and summed. The first 14 roles account for 90% of the uncovered accesses while the first 44 account for over 99% of the uncovered accesses.

ICD9 Code	# Prob = 1	Description
238.73	19	High grade myelodysplastic syndrome ...
250.71	17	Type I (juvenile type) diabetes ...
250.81	17	Type I (juvenile type) diabetes ...
191.1	16	Malignant neoplasm of frontal lobe ...
891.2	16	Open wound of knee, leg (except thigh)...
208.9	15	Unspecified leukemia
237.6	15	Neoplasm of uncertain behavior of ...
713.0	15	Arthropathy associated with other ...
781.5	15	Clubbing of fingers
228.02	14	Hemangioma of intracranial structures
287.9	14	Unspecified hemorrhagic conditions
441.3	14	Abdominal aneurysm, ruptured
753	14	Congenital anomalies of urinary system
V10.42	14	Personal history of malignant neoplasm ...

Table 6.3: **ICD-9 code coverage of mapped positions:** The number of positions for which the likelihood accessing the given diagnosis is 100%, sorted by the number of positions with that probability.

6.2 Potential Uses

One immediate application of these results is to discover the how well accesses to patients having a particular diagnosis can be explained by the user’s position. If we can map multiple NMH positions to their equivalent HPTC, then we can discover how likely a HPTC position is to access a diagnosis. This analysis can also be repeated by wrapping multiple HPTC codes into a higher level in the HPTC hierarchy.

To get an idea of how well these results can be utilized, we ask the question: What is the likelihood of a diagnosis being associated with a position code? This can be answered simply statistically analyzing our data set. For each diagnosis and HPTC, we calculate how many patients with the diagnosis are accessed by a position with the HPTC out of how many patients have the diagnosis.

It turns out that for many ICD-9 codes, the likelihood described above is very high for a large number of positions. Table 6.3 gives a list of the top 14 most well-covered ICD-9 codes by this calculation. The first column is the list of ICD-9 codes sorted by the value in the second column, which are the number of positions for which the probability calculated above is 100%. The last column gives the description of the ICD-9 code.

With the further development of the HPTC mapping, or other similar ways to derive standardized positions from the NMH positions, we should be able to better understand the structure of the data set in terms of how users access patients with certain diagnoses. This information can then be leveraged in the development of a new module which analyzes the likelihood of a user accessing a patient based on his diagnosis.

This type of module is currently implemented as the Position Explainer module, mentioned in Section

5.2, but does not utilize the standard mapping because it is not sufficiently developed at this time.

Another potential use for this mapping would be to utilize external references to the HPTC mapping. For instance, the user could be provided additional information about the position who has accessed their record, such as a website detailing the purpose of each role. This would be a significant benefit for the patient so that he or she would not be left wondering what the esoteric positions are that reference his or her account.

Chapter 7

Implementation

We will now discuss the current implementation details of the MAI system including the various technologies that are used and the ways in which the design goals set forth in Chapter 4 are achieved.

7.1 Model View Controller Architecture

The Model View Controller (MVC) architecture is implemented, broadly, as a user interface (the *front-end*), a server managing business logic (the *middleware*) and a database (the *back-end*). Patients and administrators who wish to use MAI do so by logging in to the front-end via a webpage, served by an Apache Tomcat web server running Java code. This web server contains all of the middleware which manages querying the database, organizing it, calling the analytical modules, and responding to the users' HTTP requests. The database contains a normalized data set retrieved from Northwestern Memorial Hospital of the inpatient access log over the course of 128 days.

Figure 7.5 contains a diagram of how the components are divided between front-end, middleware, and back-end and the technologies used to communicate between them.

7.1.1 User Interface

The MAI user interface is implemented as a set of JavaServer Pages (JSPs) which are compiled by the middleware before being sent to the user for viewing. The user's browser also makes asynchronous calls via JavaScript to the middleware where Java servlets reply to the requests.

The current implementation of the MAI front-end divides the user interface into two perspectives: Patient View and Administrator View. The Patient View is primarily a restricted version of the Administrator View, hiding information which is sensitive in nature, such as users or accesses which are detected as "anomalous" by the modules.


```
<coloringlist>
  <coloring>
    <userId>34357</userId>
    <doublerank>1.0</doublerank>
  </coloring>
</coloringlist>
```

Figure 7.1: Example of the coloring XML returned from the `Coloring` servlet

Administrator View

The Administrator View consists initially of a page in which the user inputs a patient ID into a text field for querying. In a real-world implementation, this would probably have multiple options for querying a user's identity as it is rare for healthcare workers to have a unique identifier for a patient. Since our dataset is de-identified, the field only asks for the `patient_id` field from the `patient` table in the database. This ID is a primary key for the `patient` table and is therefore unique.

When the `patient_id` is submitted to the form, the server responds with a new page consisting of a complete list of audit logs for that patient. This is divided into possibly several HTML tables, one for each of the patient's encounters. The user is then presented with buttons for each of the active modules currently available. If a module has encountered an error, or its preprocessing stage is incomplete, the button will not be shown.

When the user clicks a module button, an asynchronous call is made via JavaScript to the middleware by an HTTP GET request issued to the appropriate servlet. Currently the middleware implements two Java servlets, `Coloring.java` and `Descriptions.java` which handle, respectively, the tasks of coloring the access log and providing descriptions of the access log.

The HTTP GET request issued by the client-side JavaScript is sent a URL specifying the servlet to call (either `Coloring` or `Descriptions`), with the `module_id` and `patient_id` as query parameters. The `module_id` is a universally unique identifier (UUID) generated when the module is registered, and the `patient_id` is the patient's primary key in the `patient` table.

The servlet then calls the appropriate module and returns XML to the client containing the coloring or descriptions information, such as seen in Figures 7.1 and 7.2. The user's JavaScript then parses the XML and either adds coloring to the access log or generates a icons notifying the user that there is a description available. The user can then hover over the icon to read the description.

```
<descriptionlist>
  <description>
    <accessId>312546</accessId>
    <value>The user for access 312546 wrote to a note or order.</value>
  </description>
</descriptionlist>
```

Figure 7.2: Example of the description XML returned from the `Descriptions` servlet

Patient View

The Patient View for the front-end is, currently, just a restriction of the Administrator View. It is necessary to limit the amount of information presented to the patient, especially when there are potential HIPAA violations being considered.

The Patient View is implemented using the same technology as the Administrator view, but the set of modules exposed to the user are restricted. Currently, only the Position Explainer module is available to the patient because it only provides information explaining the roles which are present in the access log. The other two modules provide more detailed anomaly detection information and are therefore not given to the patient.

7.1.2 Database

The access log data in the data set was supplied by the Northwestern Memorial Hospital (NMH) in Chicago, Illinois and was gathered between late 2010 and early 2011. The dates in the access log are shifted from their actual dates of occurrence in order to comply with HIPAA de-identification rules, but the dates are accurate relative to each other. The dates in the access log span from September 4st, 2010 to January 10th, 2011. It contains over 6 million accesses divided into more than 30,000 encounters. The NMH data set was supplied as a series of flat comma-separated value (CSV) files extracted from NMH's inpatient Cerner system.

In order to make sense of such a large data set and be able to process it quickly, we wrote a set of Python scripts to read the raw CSV data and insert them into a normalized MySQL database and validate the data once it was stored.

The structure of the database itself can be examined in detail in Appendix A by viewing the entity relationship diagram and table summaries there. A number of tables not relevant to this work were excluded, namely those relating to outpatient access data.

Structure

In an effort to normalize the database, thereby minimizing its disk footprint, de-duplicating data, and improving query efficiency, many tables were constructed to hold static data which was frequently repeated.

A summary of those tables follows.

- **department**: Each user is associated with one or more departments, such as “Allergy & Immunology” and “Neurology”. There are 99 distinct departments represented in the data set.
- **icd9_data**: The data set consists of 11192 distinct International Classification of Diseases (ICD) version 9 (ICD-9) codes [27], divided between 1683 procedure codes and 9509 diagnosis codes. These codes are an internationally standardized set of codes for the purposes of uniformly identifying diseases and procedures. The **icd9_data** table consists of a textual **code** and a **type** field denoting whether it is for diagnosis or procedure. The **desc** field contains a short description of the code, such as “Aneurysm and dissection of heart.”
- **location**: This table contains the set of locations referred to in the NMH access logs. There are 57 distinct locations such as “Prentice 16” which typically have a ward- or floor-level granularity. No locations are specific enough to identify an individual room.
- **patient_service**: The patient service table contains a set of “services” to which a patient can belong. These bear a passing resemblance to the **department** table, but are not identical. Furthermore, **patient_service** elements are associated with a patient rather than with a user. Some examples include “Obstetrics” and “General Medicine.”
- **reason**: The reason values are short strings that are highly correlated to the user’s role. None of the reason values contain any contextual information about why an access is made. Examples include “ED Patient Care Staff Nurse” and “HIM.”
- **role**: This table contains the roles, or positions, of a user. Most of them are highly abbreviated and specialized names, such as “NMPG OB PEDS MD - CPOE”.

The database also consists of a number of structural tables, used to form associations between multiple tables.

- **user_dept_list**: Since each user is associated with multiple departments, this table is needed to enable such a one-to-many mapping. Each row in **user_dept_list** contains a reference to both the **user** table and the **department** tables.

- **problem**: Each patient must be associated with multiple ICD-9 diagnosis and procedure codes, so the **problem** table exists to achieve this.
- **icd9_grouping**: The ICD-9 codes form a hierarchy, wherein each ICD-9 code in the **icd9_data** table is at the bottom. The higher levels of the hierarchy are represented in the **icd9_grouping** table by referencing the first and last (inclusively) ICD-9 codes in that grouping. The table refers to itself through the **parent_id** field and provides a text description of each grouping in the **desc** field.

The **user** and **patient** tables are structured as follows.

- **user**: This table encapsulates all data about an EHR user, one of the people working in Northwestern Memorial Hospital who is recorded in the access logs. Each user has a user pseudonym that was assigned at the time the access log data was extracted, thereby uniquely identifying each user in a de-identified way. Each user has a **role_id** which maps to a position in the **role** table.
- **patient**: Like **user**, each patient has a pseudonym assigned in the same way. Each patient also has an age value and a zip code, de-identified to the initial three digits.

Lastly, the **access** and **encounter** tables contain the core information that we are interested in in this work.

- **encounter**: An encounter starts whenever a user is admitted to a clinic and ends when the the user is discharged and the billing cycle is complete. As such, each encounter has a start date (**enc_start_dttm**) and an end date (**enc_end_dttm**). It has a reference to the **patient_id** in the **patient** table for the patient of this encounter. It also has a **enc_id** which separates each encounter for a patient in the original data set.
- **access**: An access associates many entities together and is generated whenever a user accesses a patient's record within the EHR system. Each encounter is associated with a **encounter**, **patient_service**, **user**, and **reason**. There is also a date and time associated with the access in the field **user_access_dttm**. Each access also has a **user_note_cnt** and **user_order_cnt** which describe, respectively, the number of notes and orders created on this access.

7.1.3 Business Logic

The core of MAI is implemented as a series of Java classes and libraries running inside an Apache Tomcat Java container. All of the code for receiving and responding to HTTP requests, issuing queries, and launching modules is implemented in this component.

Measure	Value
Accesses	6075530
Encounters	30913
Patients	25343
Users	8341
Roles	159
Reasons	287
Departments	99
Locations	57
Patient's Services	34

Table 7.1: Statistics of the NMH access log data set

Key Classes

There are several Java classes that are critical to understanding how the MAI middleware functions.

Module: The `Module` class encapsulates the metadata about each of the modules registered on the server. This class defines enumerates for the acceptable fields in the module registration XML files: `ProcessingType`, `ParameterType`, and `ReturnType`. `PreprocessingType` can be either `None` or `DBInstance`, which determines the behavior to take at preprocessing. `ParameterType` can be any of the values listed in the first column of Table 7.2 following “parameter”. `ReturnType` currently only has the values of `Double0To1` and `Description`. The former states that the module returns a double value between 0.0 and 1.0, which can then be mapped into a color. The latter provides a brief description of the results that the module found.

`Module` also has three nested static classes which encapsulate parts of the metadata within the registration XML. `PreprocessingMeta` describes the preprocessing information, `InitializationMeta` gives information about what is required when the server initializes a new user session, and `RuntimeQueryMeta` stores information about the inputs and outputs expected when processing a query. `InitializationMeta` is currently used to bundle multiple queries into a single batch processing via the `BatchProcess` interface.

ModuleRegistry: As its name suggests, the `ModuleRegistry` is responsible for maintaining a collection of modules which are registered with the server. It is a singleton class so that there is only a single instance of the `ModuleRegistry` present on the server’s virtual machine at any time. The `ModuleRegistry` generates a universally unique identifier (UUID) for each module registered with the middleware, so that the front-end can have an unambiguous means to reference a module.

DBConnectionManager: This class is a thin wrapper around a JDBC instance which facilitates querying the database for the module developer. It is instantiated by the MAI middleware with read-only capabilities to prevent any unwanted changes to the database. This object is passed to every module via the `setConnectionManager()` method required by the `ModuleInterface` class. This interface is described in more detail in Section 7.2.2.

Bootstrap Process

When the Tomcat server is started, a bootstrap thread is launched in order to initialize the server. At this point, the server instantiates the `ModuleRegistry` and loads the module definitions from their XML sources as outlined in Section 7.3.

Once all of the `Module` objects have been generated from the XML, their structure is validated. The server verifies that the class referenced in the registration XML can be located and implements all of the required interfaces based on their declared capabilities. This is done according to the specification shown in Table 7.2.

After the structure of the `ModuleRegistry` has been validated, the bootstrap thread initializes the `DBConnectionManager` for each module and passes it through the `ModuleInterface.setConnectionManager()` method. The middleware then calls the `ModuleInterface.preprocess()` method to begin module preprocessing. The server quickly checks to see if the preprocessing is complete by calling `ModuleInterface.preprocessingComplete()` so that modules without preprocessing can be set to a complete state immediately.

HTTP Responses

After the bootstrap thread finishes, the middleware is ready to begin serving clients who log on via the MAI webpage. A thin shell of the webpage is implemented through JavaServer Pages. These pages are compiled into Java bytecode when a client issues an HTTP request, and the execution of that bytecode results in a standard HTML webpage.

Based on the user's actions one of three pages is displayed: the welcome page, the administrator query page, and the patient query page. The welcome page is a screen asking for the user to either select the administrator or patient views, or to issue a query for a `patient_id`. When a query is submitted from the welcome page, it defaults to the administrator view.

When the user inputs a `patient_id` query, a helper class called `IndexHelper` is instantiated with the HTTP request data. This helper class then handles all of the logic behind what content is generated and forwarded back to the user as HTML. The `IndexHelper` issues the query in Figure 7.3 against the back-end looking for all accesses to this patient. This query finds all encounters belonging to the current patient

```

select a.access_id , a.user_access_dttm , a.user_id , r.role_desc , a.enc_uniq_id
from access a inner join user u on a.user_id = u.user_id
inner join role r on r.role_id = u.role_id
inner join encounter e on e.enc_uniq_id = a.enc_uniq_id
where e.patient_id = <patient_id>
order by e.enc_uniq_id , a.user_access_dttm ;

```

Figure 7.3: SQL query to select all needed information about accesses for the given patient_id

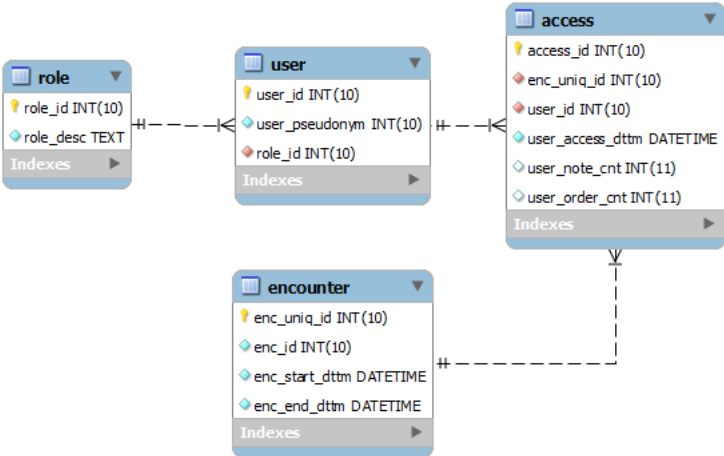


Figure 7.4: A subset of the diagram relevant to the query in 7.3

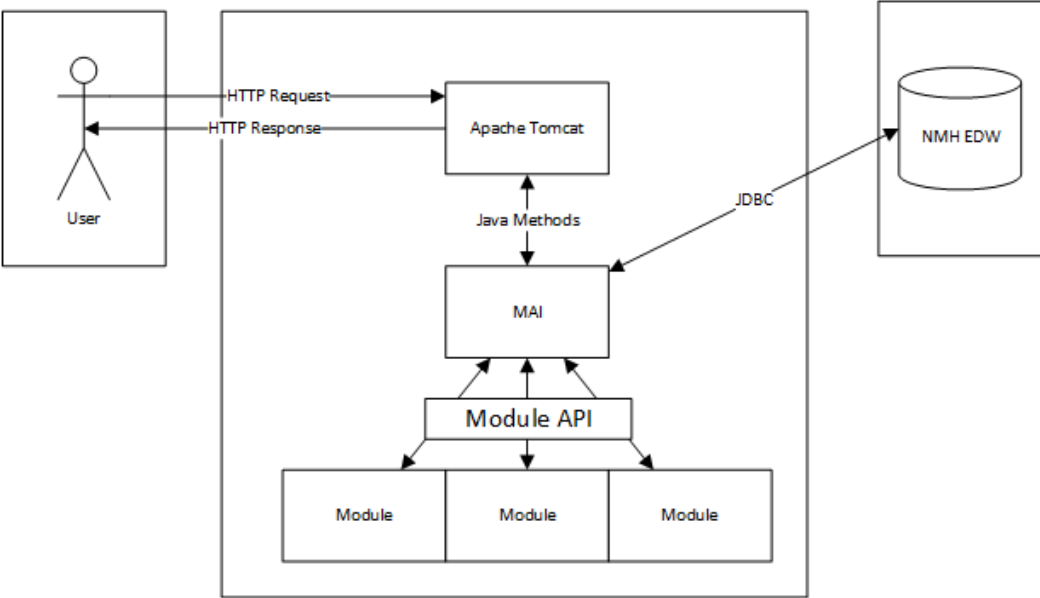


Figure 7.5: **Architecture Component Detail.** The large boxes are, from left to right, the front-end, middleware, and back-end. The front-end communicates with the middleware through HTTP while Tomcat listens for incoming connections. Once a request is received it is forwarded to the MAI Java code. MAI communicates with the registered modules via a series of Java Interfaces and supplies a read-only database connection to the modules.

Capability	Java interfaces	Optional?	Phase when called
Any	ModuleInterface.java	N	Module initialization
parameter/AccessId	NeedsAccessId.java	N	Query Initialization/Runtime Queries
parameter/UserId	NeedsUserId.java	N	Query Initialization/Runtime Queries
parameter/RoleId	NeedsRoleId.java	N	Query Initialization/Runtime Queries
parameter/PatientId	NeedsPatientId.java	N	Query Initialization/Runtime Queries
parameter/Timestamp	NeedsTimestamp.java	N	Query Initialization/Runtime Queries
returnType/Description	ReturnsDescription.java	N	Runtime Queries
returnType/Double0To1	ReturnsType.java	N	Runtime Queries
returnType/Double0To1	BatchProcess.java	Y	Runtime Queries

Table 7.2: **Capabilities and Interfaces:** The capabilities of a module and the required Java interfaces to implement them. The first column contains the capability, which are divided into their semantic meaning, i.e. whether they are a parameter or return type. Items marked “parameter/X” mean that X is a required parameter and items marked “returnType/X” mean that X is returned by the module. In both cases X must be declared in the registration XML. The second column gives the Java interface to implement and the third column states whether it is optional to implement the interface in order to have that capability. The last column gives the lifetime phase(s) in which that interface is used.

7.2 Module API Implementation

7.2.1 The Module Lifecycle

The module lifecycle is heavily controlled by the information that is defined in each module’s registration XML. Examples of these XML files can be found in Figures 7.8, 7.9 and 7.9. When a module is registered, the `ModuleRegistry` organizes all of the parameters required in the following phases of the module’s lifecycle.

During Phase 2, preprocessing, the module is given a database instance and told to begin preprocessing. When the module begins declaring that its preprocessing is complete, the module buttons become visible to the front-end user and the module enters Phase 3, run-time queries (RTQ).

Each time a user queries for a patient on the front-end, the module is given the type of data it declares as a `parameter` in the `initializationMeta` element. The purpose of initialization metadata is mostly to yield more efficient queries and computations which assure a more responsive user interface. Then the middleware gives the module the parameters it requests in the `runtimeQueryMeta` element.

Phase 4, uninstallation, is not currently implemented.

7.2.2 Module Java Interfaces

When a module capability is declared in the registration XML file, the accompanying Java interfaces must be implemented within the interface for that module so that the middleware can call the appropriate methods. A summary of which Java interfaces go with each capability can be located in Table 7.2. The interface hierarchy is shown in Figure 7.6.

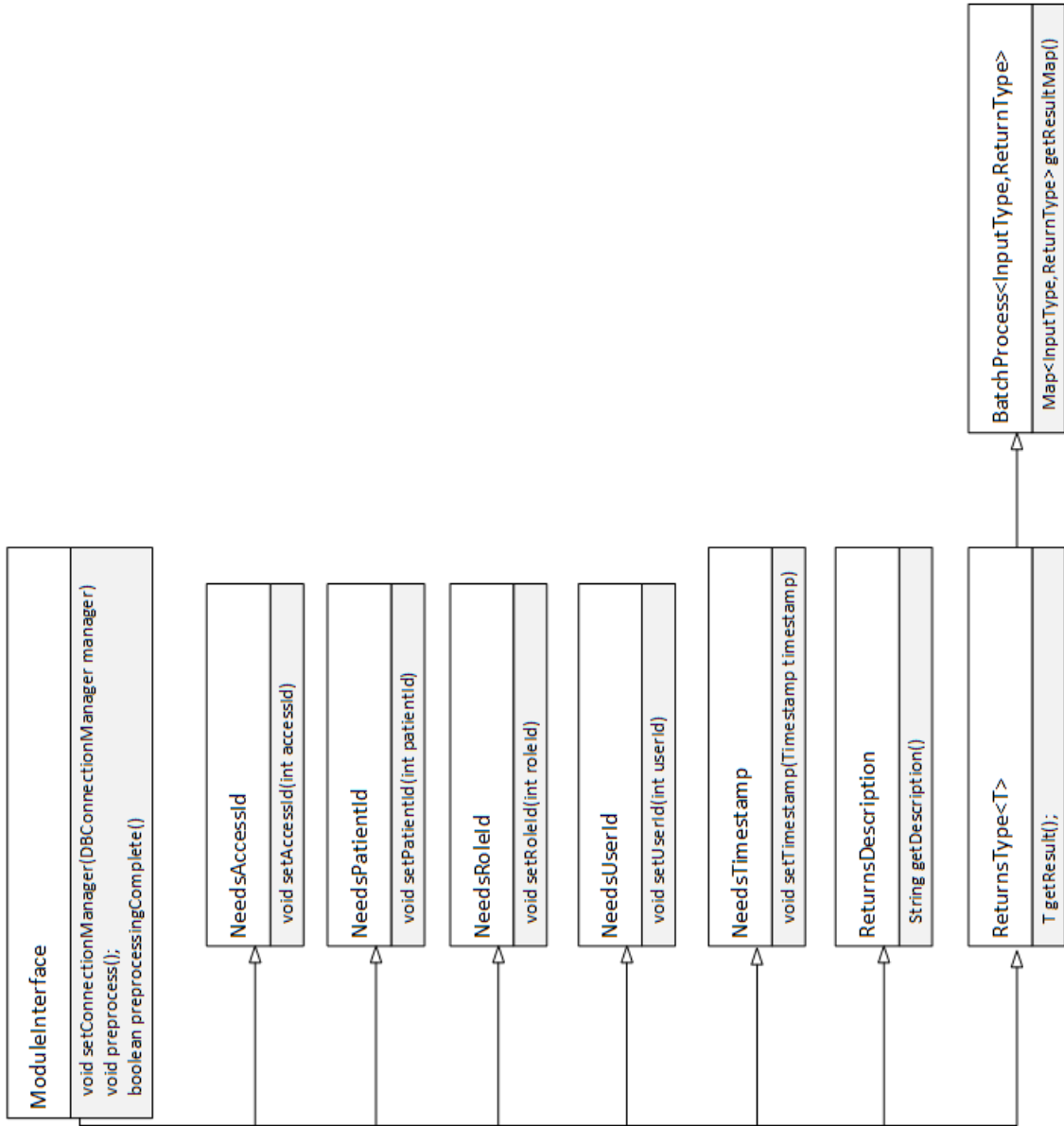


Figure 7.6: Current Module API Java Interfaces Hierarchy

```

select a.access_id from encounter e
inner join access a on a.enc_uniq_id = e.enc_uniq_id
where e.patient_id = <patient_id>
and (user_order_cnt > 0 or user_note_cnt > 0)

```

Figure 7.7: Query to retrieve Notes and Orders data

7.3 Module Implementations

7.3.1 Notes and Orders

The Notes and Order module is implemented entirely in Java. The Notes and Orders module class directly implements four Java interfaces: `NeedsPatientId`, `NeedsAccessId`, `BatchProcess<Integer,Double>`, and `ReturnsDescription`. It implicitly implements `ModuleInterface` and `ReturnsType<Double>` since `ReturnsType<Double>` is a superinterface of `BatchProcess<Integer,Double>` and `ModuleInterface` is a superinterface of all the Module API Java interfaces. The module registration XML can be seen in Figure 7.8.

Since Notes and Orders does not require any preprocessing, the `ModuleInterface.preprocess()` method is blank and `ModuleInterface.preprocessingComplete()` returns `true` immediately.

The registration XML states that to initialize the `ModuleInterface` for each query session, the `patient_id` should be provided. This allows the Notes and Orders module to retrieve all of the access data for the given patient within a single query, rather than depending upon the middleware to send each `access_id` one at a time. When initialization occurs, the Notes and Orders module then issues the query in Figure 7.7 to retrieve the patient data.

When performing batch processing, the `BatchProcess<Integer,Double>`. `getResultMap()` function returns a map associating `access_ids` with a double value score. If an access appears in the query in Figure 7.7 then the score is set to a constant value representing a non-suspicious score. When an access does not result from the above query, its score is omitted and assumed to be neutral.

7.3.2 Position Explainer

Like Notes and Order, the Position Explainer module is implemented entirely in Java and does not require any preprocessing, though its performance would be improved if it were implemented. Position Explainer is not like the other two modules in that it does not return any anomaly detection results, but rather only explains the accesses that occur in the log. It does this in the following way:

```

<?xml version="1.0" encoding="utf-8"?>
<module>
  <name>Notes and Orders</name>
  <class>seclab.aod.module.NotesAndOrdersModule</class>
  <location>/NotesAndOrdersModule.class</location>
  <preprocessingMeta>
    <dataFormat>None</dataFormat>
  </preprocessingMeta>
  <initializationMeta>
    <parameters>
      <parameter>PatientID</parameter>
    </parameters>
  </initializationMeta>
  <runtimeQueryMeta>
    <parameters>
      <parameter>AccessID</parameter>
    </parameters>
    <returnTypes>
      <returnType>Double0To1</returnType>
      <returnType>Description</returnType>
    </returnTypes>
  </runtimeQueryMeta>
</module>

```

Figure 7.8: Registration XML for the Notes and Orders module.

- Issue a query for all diagnoses for this user.
- For each diagnosis, query for how many patients with that diagnosis are in the entire access log.
- Query for how many patients are accessed by that role in total.
- Combine the last two values into a probability, i.e. find how likely a role is to access that diagnosis.
- If the resulting probability is greater than some threshold, report it to the user.

Note that this module could easily be also implemented as an anomaly-detection module by using the reported probabilities to determine an anomaly score. The registration XML for this module is available in 7.9.

7.3.3 CADS

The CADS module is by far the most complex of the three modules implemented so far. CADS requires a considerable amount of preprocessing time in order to run on the entire access log, regardless of the k value it is given. It also requires persistent storage on the machine on which it is implemented. There are also a number of ways in which to use the results that it returns to assign anomaly scores.

```

<?xml version="1.0" encoding="utf-8"?>
<module>
  <name>Explain Positions</name>
  <class>seclab.aod.module.PositionExplainer</class>
  <location>/PositionExplainer.class</location>
  <preprocessingMeta>
    <dataFormat>DBInstance</dataFormat>
  </preprocessingMeta>
  <initializationMeta>
    <parameters>
      <parameter>PatientID</parameter>
    </parameters>
  </initializationMeta>
  <runtimeQueryMeta>
    <parameters>
      <parameter>UserId</parameter>
    </parameters>
    <returnTypes>
      <returnType>Description</returnType>
    </returnTypes>
  </runtimeQueryMeta>
</module>

```

Figure 7.9: Registration XML for the Position Explainer module.

```

select e.patient_id, a.user_id from encounter e
inner join access a on e.enc_uniq_id = a.enc_uniq_id
order by a.user_access_dttm

```

Figure 7.10: SQL query issued to construct the entire access log for CADs

The CADS code itself was implemented in the R programming language by the authors of [7]. Executing the CADS module therefore requires the R environment to be installed and configured on the server. When the preprocessing stage is reached, the CADS module retrieves a stripped-down form of the access log for processing. The input to the CADS R code is a series of three vectors in a specific format. Each vector is length N where N is the number of distinct accesses in the access log. The first vector represents the users, the second is the patients, and the third are the number of times the i^{th} user accessed the i^{th} patient's record. The values in the user and patient vectors are simply unique identifiers which are sequential between 1 and the maximum number of users or patients, respectively.

The access log is gathered by executing the SQL query in Figure 7.10 with optional limitations such as setting the maximum number of records retrieved or setting the start and end dates and times. Once the patient and user identifiers are gathered, they are then mapped to sequential values beginning at 1 and saved to disk. The CADS module then constructs the R code from the script source and tells R to load each of the values from disk and execute the CADS script. The output is published to an XML file in a temporary directory and read back in through Java.

Once the CADS scores are generated, there are two options currently implemented in the CADS module for converting these scores to anomaly values. The first approach is to divide the scores into percentiles of some given size and take the topmost partition based on that percentile. With this approach you are guaranteed to get the top $p\%$ of scores for a given percentile p . The better approach, which is the default, is to calculate the standard deviation and mean on the deviations returned by CADS and assign anomaly scores to users based on their distance from the mean. Both of these approaches are performed by inserting R code into the script and having R execute the statistical analysis, rather than implementing it in Java.

The current implementation of the CADS module using a k value of 10 and a 100,000 record subset of the entire access log. It uses the normal distribution method for determining anomalous users with a 2 standard deviation threshold. When it finds a user having a CADS score 2 standard deviations away from the mean, it assigns a color to that user based on the z-score. This way, users who are further from the mean will appear a deeper red than those who are closer to the mean. All users who fall within the 2 standard deviation threshold are not colored.

The CADS module also produces a description of the coloring for all users who lie outside of the threshold, which alert the front-end user to the reason why the user was colored red.

```

<?xml version="1.0" encoding="utf-8"?>
<module>
  <name>CADS</name>
  <class>seclab.aod.module.CADSMODULE</class>
  <location>/CADSMODULE.class</location>
  <preprocessingMeta>
    <dataFormat>DBInstance</dataFormat>
  </preprocessingMeta>
  <initializationMeta>
    <parameters>
      <parameter>PatientID</parameter>
    </parameters>
  </initializationMeta>
  <runtimeQueryMeta>
    <parameters>
      <parameter>UserID</parameter>
    </parameters>
    <returnTypes>
      <returnType>Double0To1</returnType>
      <returnType>Description</returnType>
    </returnTypes>
  </runtimeQueryMeta>
</module>

```

Figure 7.11: Registration XML for the CADS module.

Chapter 8

Demonstration and Use Case

This chapter will present an example scenario in which MAI is utilized by both a patient and administrators. We hope to demonstrate the power of MAI in satisfying the requirements of patients and administrators alike, and give a taste of what can be had once more modules are developed and MAI becomes more than a prototype.

8.1 Patient View

Imagine that you were recently a patient in a hospital who has undergone treatment for issues related to your internal organs. You were given the `patient_id` of 35347 and diagnoses in Table 8.1 during your five encounters at the hospital.

You remember there being a lot of people coming in and out of your hospital room, so you become curious to know who all these people were and why they need access to your sensitive personal information. You know that you have the right to an access report, so you file a request with the provider, who are obligated by law to give it to you.

The provider responds telling you to log into their MAI web page using your `patient_id`. You log on to the page and are greeted with the welcome screen, as in Figure 8.1.

You click on patient view and enter your `patient_id` in the search field, which produces a list of your accesses. In our de-identified prototype, all users have an ID rather than a name displayed. Since you were accessed by 319 distinct users over the course of your 5 encounters, the access record is very long and contains a large number of unfamiliar user positions, such as “UR/QA 1”.

According to the requirements of the U.S. Department of Health and Human Services, your record

ICD-9 Code	Description
153.9	Malignant neoplasm of colon, unspecified site
197.0	Secondary malignant neoplasm of lung
197.7	Secondary malignant neoplasm of liver

Table 8.1: Summary of the diagnoses for patient 35347

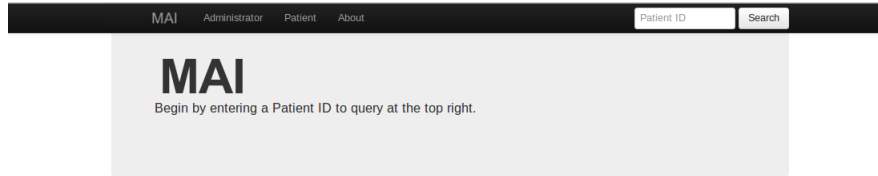


Figure 8.1: The MAI welcome page

displays the date and time of access, the name of the users who made the access, and the user's role which is used in lieu of a reason or action taken on the data.

Additionally there is a button reading "Explain Positions" at the top of the page. When you press it, each user has a small question mark icon next to their name. When you hover over the icon for the user with position UR/QA 1, a tool-tip text appears reading,

This role has a 80.0% chance of accessing your diagnosis, 197.0

This role has a 82.0% chance of accessing your diagnosis, 153.9

This role has a 93.8% chance of accessing your diagnosis, 197.7

An example of this functionality is seen in Figure 8.2.

You begin to feel a lot better about seeing strange positions that you previously weren't aware of. However, since you had so many accesses to your record, you decide to call your healthcare provider and demand that they audit your record for any anomalous behavior.

8.2 Administrator View

The security administrator at your provider receives your request for an audit and proceeds to check your record through the MAI system. When she logs in with your `patient_id`, she sees the same access log that you did when you logged in. However, she has several modules available to her that you did not have access to. In this demonstration, these are the Notes and Orders module and the CADS module. A screenshot of the administrator page can be found in Figure 8.3.

MAI	Administrator	Patient	About	Patient ID	Search
5	Wednesday, September 29, 2010 06:24 PM	14580	👤	Patient Care Staff Nurse	
6	Wednesday, September 29, 2010 06:27 PM	96289	👤	Patient Care Staff Nurse	
7	Wednesday, September 29, 2010 06:29 PM	96289	👤	Patient Care Staff Nurse	
8	Wednesday, September 29, 2010 06:49 PM	96289	👤	Patient Care Staff Nurse	
9	Wednesday, September 29, 2010 06:59 PM	14580	👤	Patient Care Staff Nurse	
10	Wednesday, September 29, 2010 07:02 PM	20173	👤		This role has a 80.0% chance of accessing your diagnosis, 197.0 This role has a 87.5% chance of accessing your diagnosis, 153.9 This role has a 98.4375% chance of accessing your diagnosis, 197.7
11	Wednesday, September 29, 2010 07:05 PM	96289	👤	Patient Care Staff Nurse	
12	Wednesday, September 29, 2010 07:08 PM	13683	👤	Patient Care Assistive Staff	
13	Wednesday, September 29, 2010 07:08 PM	13683	👤	Patient Care Assistive Staff	
14	Wednesday, September 29, 2010 07:10 PM	13683	👤	Patient Care Assistive Staff	
15	Wednesday, September 29, 2010 07:15 PM	96289	👤	Patient Care Staff Nurse	
16	Wednesday, September 29, 2010 07:20 PM	13683	👤	Patient Care Assistive Staff	
17	Wednesday, September 29, 2010 07:21 PM	13683	👤	Patient Care Assistive Staff	
18	Wednesday, September 29, 2010 07:23 PM	96289	👤	Patient Care Staff Nurse	
19	Wednesday, September 29, 2010 07:24 PM	96289	👤	Patient Care Staff Nurse	
	Wednesday, September 29, 2010 07:26				

Figure 8.2: The PositionExplainer module at work

MAI	Administrator	Patient	About	Patient ID	Search
MAI					
Patient ID: 35347					
Notes and Orders Explain Positions CADS					
Encounter: 1866					
Row	Date/Time	User		Role	
1	Wednesday, September 29, 2010 05:49 PM	64790		RAD - Nurse	
2	Wednesday, September 29, 2010 05:51 PM	86		RAD - Resident/Fellow	
3	Wednesday, September 29, 2010 06:09 PM	819994		RAD - Technologist	
4	Wednesday, September 29, 2010 06:14 PM	15449		Reg/Sched Super-User	
5	Wednesday, September 29, 2010 06:24 PM	14580		Patient Care Staff Nurse	
6	Wednesday, September 29, 2010 06:27 PM	96289		Patient Care Staff Nurse	
7	Wednesday, September 29, 2010 06:29 PM	96289		Patient Care Staff Nurse	
8	Wednesday, September 29, 2010 06:49 PM	96289		Patient Care Staff Nurse	
9	Wednesday, September 29, 2010 06:59 PM	14580		Patient Care Staff Nurse	
10	Wednesday, September 29, 2010 07:02 PM	20173		NMH Physician Hospitalist-CPOE	
11	Wednesday, September 29, 2010 07:05 PM	96289		Patient Care Staff Nurse	
12	Wednesday, September 29, 2010 07:08 PM	13683		Patient Care Assistive Staff	
13	Wednesday, September 29, 2010 07:08 PM	13683		Patient Care Assistive Staff	
14	Wednesday, September 29, 2010 07:10 PM	13683		Patient Care Assistive Staff	
15	Wednesday, September 29, 2010 07:15 PM	96289		Patient Care Staff Nurse	
16	Wednesday, September 29, 2010 07:20 PM	13683		Patient Care Assistive Staff	
17	Wednesday, September 29, 2010 07:21 PM	13683		Patient Care Assistive Staff	
18	Wednesday, September 29, 2010 07:23 PM	96289		Patient Care Staff Nurse	
19	Wednesday, September 29, 2010 07:24 PM	96289		Patient Care Staff Nurse	
20	Wednesday, September 29, 2010 07:26 PM	96289		Patient Care Staff Nurse	
21	Wednesday, September 29, 2010 07:37 PM	6667		NMH Physician Hospitalist-CPOE	

Figure 8.3: The MAI administrator display page


	MAI	Administrator	Patient	About	Patient ID	Search
89	Thursday, September 30, 2010 02:56 PM	29320	Advanced Practice Clinician - CPOE			
90	Thursday, September 30, 2010 04:23 PM	6670	HIM			
91	Thursday, September 30, 2010 04:32 PM	14606	RAD - Nurse			
92	Thursday, September 30, 2010 06:06 PM	86	RAD - Resident/Fellow			
93	Thursday, September 30, 2010 09:51 PM	6695	NMH Physician-CPOE			
94	Friday, October 01, 2010 06:29 AM	86	RAD - Resident/Fellow			
95	Friday, October 01, 2010 07:33 AM	6567	Advanced Practice Clinician - CPOE			
96	Friday, October 01, 2010 02:47 PM	6670	HIM			
97	Friday, October 01, 2010 02:49 PM	6670	HIM			
98	Saturday, October 02, 2010 08:37 AM	6679	NMH Physician Office - CPOE			
99	Saturday, October 02, 2010 12:09 PM	86	 User 6679 has a high social network-derived score			
100	Saturday, October 02, 2010 12:25 PM	48110	ED Patient Care Staff Nurse			
101	Saturday, October 02, 2010 12:28 PM	713987	ED Patient Care Staff Nurse			
102	Saturday, October 02, 2010 12:36 PM	67793	ED NMH Physician-CPOE			
103	Saturday, October 02, 2010 01:16 PM	85	NMH Resident/Fellow-CPOE			
104	Saturday, October 02, 2010 01:16 PM	85	NMH Resident/Fellow-CPOE			
105	Saturday, October 02, 2010 01:29 PM	85	NMH Resident/Fellow-CPOE			
106	Saturday, October 02, 2010 01:31 PM	713987	ED Patient Care Staff Nurse			
107	Saturday, October 02, 2010 01:31 PM	35248	ED NMH Physician-CPOE			
108	Saturday, October 02, 2010 01:40 PM	67793	ED NMH Physician-CPOE			
109	Saturday, October 02, 2010 02:34 PM	67793	ED NMH Physician-CPOE			
110	Saturday, October 02, 2010 02:35 PM	67793	ED NMH Physician-CPOE			
111	Saturday, October 02, 2010 03:24 PM	84	ED Patient Care Staff Nurse			
112	Saturday, October 02, 2010 03:39 PM	84	ED Patient Care Staff Nurse			
113	Saturday, October 02, 2010 03:41 PM	713987	ED Patient Care Staff Nurse			

Figure 8.4: A lightly-colored user affected by the CADS module

The administrator begins examining the record by selecting the CADS module, which colors a few users in various shades of red and places a question mark icon next to the names of those users. The administrator hovers her mouse over the icon for user 6679, who is shaded in a faint red, like in Figure 8.4. The tool-tip text appears, displaying

User 6679 has a high social network-derived score

which indicates that CADS has deemed user 6679 suspicious based on its configuration parameters. She decides that there is no need for alarm yet, since the CADS module was configured to begin shading users red when their deviations are outside of two standard deviations from the mean. She continues searching for users who have a higher alert level.

Before long, she discovers user 6736, who has a significantly deeper red color than the previous user, as seen in Figure 8.5. She begins to get suspicious of this user, so she enables the Notes and Orders module, which colors a large number of the accesses green, indicating that they are not likely to be suspicious.

User 6736 also has a number of his or her accesses turn green at that point, but the red color enabled by CADS remains. She also enables the Position Explainer module which gives statistics about the user-diagnosis relationship, as in Figure 8.6. The Position Explainer module says that the greatest chance of user

MAI				Administrator	Patient	About	Patient ID	Search
472	Tuesday, November 16, 2010 07:19 AM	6709	Rehab PT					
473	Tuesday, November 16, 2010 07:20 AM	22089	Patient Care Staff Nurse (Pilot)					
474	Tuesday, November 16, 2010 07:20 AM	17361	Patient Care Assistive Staff					
475	Tuesday, November 16, 2010 07:21 AM	6736	NMH Physician Hospitalist-CPOE					
476	Tuesday, November 16, 2010 07:22 AM	6736	NMH Physician Hospitalist-CPOE					
477	Tuesday, November 16, 2010 07:29 AM	6736	NMH Physician Hospitalist-CPOE					
478	Tuesday, November 16, 2010 07:30 AM	6736	NMH Physician Hospitalist-CPOE					
479	Tuesday, November 16, 2010 07:30 AM	22089	Patient Care Staff Nurse (Pilot)					User 6736 has a high social network-derived score
480	Tuesday, November 16, 2010 07:31 AM	6736	NMH Physician Hospitalist-CPOE					
481	Tuesday, November 16, 2010 07:31 AM	11448	Patient Care Staff Nurse					
482	Tuesday, November 16, 2010 07:31 AM	22089	Patient Care Staff Nurse (Pilot)					
483	Tuesday, November 16, 2010 07:32 AM	22089	Patient Care Staff Nurse (Pilot)					
484	Tuesday, November 16, 2010 07:33 AM	11419	Unit Secretary 2					
485	Tuesday, November 16, 2010 07:53 AM	11866	Patient Care Assistive Staff					
486	Tuesday, November 16, 2010 07:53 AM	11866	Patient Care Assistive Staff					
487	Tuesday, November 16, 2010 07:54 AM	11419	Unit Secretary 2					
488	Tuesday, November 16, 2010 07:57 AM	15981	PathNet: Gen Lab Coordinator					
489	Tuesday, November 16, 2010 08:01 AM	6736	NMH Physician Hospitalist-CPOE					
490	Tuesday, November 16, 2010 08:01 AM	22089	Patient Care Staff Nurse (Pilot)					
491	Tuesday, November 16, 2010 08:13 AM	6736	NMH Physician Hospitalist-CPOE					
492	Tuesday, November 16, 2010 08:21 AM	37452	NMH Physician Office - CPOE					
493	Tuesday, November 16, 2010 08:22 AM	11448	Patient Care Staff Nurse					
494	Tuesday, November 16, 2010 08:28 AM	11448	Patient Care Staff Nurse					
495	Tuesday, November 16, 2010 08:32 AM	11448	Patient Care Staff Nurse					
496	Tuesday, November 16, 2010 08:33 AM	11991	Rehab PT					
497	Tuesday, November 16, 2010 08:38 AM	12448	NMH Resident/Fellow-CPOE					
498	Tuesday, November 16, 2010 08:41 AM	6736	NMH Resident/Fellow ID Clinic-					

Figure 8.5: A darkly-colored user affected by the CADS module

6736's role accessing a patient with this diagnosis is 75%. This means that users with the same role as user 6736 accessed patients having one of the same diagnoses as the patient 3 times out of 4. Since one of the accesses is colored green it is quite possible that the user is doing legitimate work, but he or she also makes several accesses that are not accompanied by notes or orders. The administrator decides that she should probably take a look into the accesses made by user 6736 to other patients to see if any other suspicious behavior sticks out.

8.3 Use Case Discussion

By utilizing the MAI framework and its ever-growing collection of analytical modules, the hospital was able to save a great deal of time, money, and talent in order to comply with the regulations of the Department of Health and Human services. In this example, the MAI system offered the following benefits:

- First, the MAI system automatically compiles an access record ready for presentation to the patient which eliminates time spent generating the data by specialists.
- Secondly, the patient is presented with options for explaining the accesses that are made to his or her

MAI				Administrator	Patient	About	Patient ID	Search
468	Tuesday, November 16, 2010 07:00 AM	67872	NMH Resident/Fellow-CPOE					
469	Tuesday, November 16, 2010 07:00 AM	22089	Patient Care Staff Nurse (Pilot)					
470	Tuesday, November 16, 2010 07:18 AM	22089	Patient Care Staff Nurse (Pilot)					
471	Tuesday, November 16, 2010 07:18 AM	17361	Patient Care Assistive Staff					
472	Tuesday, November 16, 2010 07:19 AM	6709	Rehab PT					
473	Tuesday, November 16, 2010 07:20 AM	22089	Patient Care Staff Nurse (Pilot)					
474	Tuesday, November 16, 2010 07:20 AM	17361	Patient Care Assistive Staff					
475	Tuesday, November 16, 2010 07:21 AM	6736	NMH Physician Hospitalist-CPOE					
476	Tuesday, November 16, 2010 07:22 AM	673	NMH Physician Hospitalist-CPOE					This role has a 70.0% chance of accessing your diagnosis, 197.0 This role has a 50.0% chance of accessing your diagnosis, 153.9 This role has a 75.0% chance of accessing your diagnosis, 197.7
477	Tuesday, November 16, 2010 07:29 AM	6736	NMH Physician Hospitalist-CPOE					
478	Tuesday, November 16, 2010 07:30 AM	6736	NMH Physician Hospitalist-CPOE					
479	Tuesday, November 16, 2010 07:30 AM	22089	Patient Care Staff Nurse (Pilot)					
480	Tuesday, November 16, 2010 07:31 AM	6736	NMH Physician Hospitalist-CPOE					
481	Tuesday, November 16, 2010 07:31 AM	11448	Patient Care Staff Nurse					
482	Tuesday, November 16, 2010 07:31 AM	22089	Patient Care Staff Nurse (Pilot)					

Figure 8.6: A combination of CADS, Notes and Orders, and Position Explainer modules

record. This behavior is likely to prevent patients from becoming confused by, or concerned about, who makes accesses to their record.

- Thirdly, a significant amount of manpower was saved in replying to a patient's request for an audit by leveraging existing anomaly detection techniques in a user-friendly way. Although providers are not required to respond to patient audit requests, doing so is good for keeping the patient satisfied and ensuring the patient returns to the same provider. Furthermore, it has the potential to save on legal liabilities if a HIPAA violation is discovered.

Chapter 9

Discussion

In this text we have outlined the need for an extensible system capable of analyzing access log data in an electronic health record environment and explored the reality of building such a system. Due to the nature of the legal and regulatory systems surrounding the United States healthcare industry, it has become necessary to build a system like MAI in order to satisfy the needs of patients and healthcare providers alike. MAI was designed with the goal in mind of easing the burden of implementation of these regulations while simultaneously leveraging the wealth of technical knowledge available in the academic literature.

9.1 Affects on Healthcare Information Technology

We hope that this study reinforces the notion that implementation of the access report requirements in the Department of Health and Human Services' Notice of Proposed Rulemaking are quite tractable. When the Office of Civil Rights called for comments on the NPRM from stakeholders, it was largely endorsed and applauded for its advancement of patient rights [26][25][4]. However, these same sources were doubtful about the capabilities of healthcare organizations to produce the access reports due to the administrative overhead required as well as the cost of implementation. We have demonstrated that compiling the access reports in a user-friendly manner is tractable in real-time for an audit log of over six million entries, and we have demonstrated that the state of the art is sufficiently advanced to go further than the requirements of the NPRM in extracting explanations for these accesses. Our prototype allows patients and administrators to be able to easily obtain information and make judgements about a complex system without the need to understand the implementation details.

The access records that HHS has called for are a valuable step toward giving patients control over the data contained within their health records. We would like to see technology driving the way to progress the way it has been demonstrated in the HITECH bill and we hope that this work stimulates more work in that direction.

9.2 Limitations and Future Work

We believe that MAI is sufficiently extensible to facilitate new detection algorithms in the future which discover more data types than simply an anomaly score or a description text. We also see great potential in the flexibility of the Module API for allowing new modules to be developed in emerging technologies.

The prototype described in these pages is fairly limited in its power to address the concerns of its users, mostly due to the lack of analytical modules that are implemented so far. Furthermore, there is a much broader limitation to this project which lies in the inherent complexity of EHR systems and our lack of understanding of it. There are no perfect algorithms for understanding the accesses that are made to a patient's record and the anomaly detection tools that are developed by the machine learning community are always somewhat handicapped by a trade-off between sensitivity and specificity.

We already have two other modules being implemented in the MAI system which will further improve its analytical capabilities. The first, called PFADS, analyzes the flow of patients through an organization according to the access records and constructs a model for a typical patient of that type [30]. A new patient can then be compared to the existing patient flows and if an access deviates from the expected model, that access can be flagged as anomalous. The second module in development, SNAD, constructs a social network similar to that in CADS but performs better at deeming specific actions as anomalous, rather than aggregate behavior like CADS [8].

We also have plans to include more information for the Patient View, such as utilizing social networks for the opposite purpose as in CADS. Rather than detecting when a user is anomalous within a community, we could use these networks to defined when a user is considered normal within his or her network based on criteria such as who he or she works with. If a patient regularly interacts with User A and User A regularly works with User B, it should not necessarily be suspicious if User B accesses the patient's record even if it is unusual for his role.

References

- [1] Emoat: Extensible medical open audit toolkit. <http://hiplab.mc.vanderbilt.edu/projects/emoat/>.
- [2] Fair warning. <http://www.fairwarning.com/>.
- [3] P2sentinel. <https://store.cerner.com/items/1552>.
- [4] American Hospital Association (AHA). Remark on nprm. <http://www.aha.org/advocacy-issues/letter/2011/110801-cl-hipaaprivreruleacctdiscl.pdf>.
- [5] American Medical Association. Hipaa 101: How it started and whats next. <http://www.ama-assn.org/ama1/pub/upload/mm/399/hipaa-101-fact-sheet.pdf>, March 2011.
- [6] Twila Brase. Policy insights. *Citizens' Council for Health Freedom - Government Health Surveillance*, 1, July 2012.
- [7] You Chen and Bradley Malin. Detection of anomalous insiders in collaborative environments via relational analysis of access logs. In *Proceedings of the first ACM conference on Data and application security and privacy*, pages 63–74. ACM, 2011.
- [8] You Chen, Steve Nyemba, Wen Zhang, and Bradley Malin. Leveraging social networks to detect anomalous insider actions in collaborative environments. In *Intelligence and Security Informatics (ISI), 2011 IEEE International Conference on*, pages 119–124. IEEE, 2011.
- [9] Daniel Fabbri and Kristen Lefevre. Explaining accesses to electronic medical records using diagnosis information. *Journal of the American Medical Informatics Association, Special Focus Issue on Biomedical Data Privacy*, 2012.
- [10] Centers for Disease Control and Prevention. Meaningful use - introduction. <http://www.cdc.gov/ehrmeaningfuluse/introduction.html>.
- [11] Centers for Medicare and Medicaid Services. Meaningful use. http://www.cms.gov/Regulations-and-Guidance/Legislation/EHRIncentivePrograms/Meaningful_Use.html.
- [12] Centers for Medicare and Medicaid Services. Taxonomy. <http://www.cms.gov/Medicare/Provider-Enrollment-and-Certification/MedicareProviderSupEnroll/Taxonomy.html>.
- [13] Centers for Medicare and Medicaid Services. Stage 1 vs. stage 2 comparison table for eligible hospitals and cahs. <http://www.cms.gov/Regulations-and-Guidance/Legislation/EHRIncentivePrograms/Downloads/Stage1vsStage2CompTablesforHospitals.pdf>, August 2012.
- [14] Centers for Medicare and Medicaid Services. Stage 2 overview tipsheet. http://www.cms.gov/Regulations-and-Guidance/Legislation/EHRIncentivePrograms/Downloads/Stage2Overview_Tipsheet.pdf, August 2012.
- [15] Robert Wood Johnson Foundation, George Washington University Medical Center, and Institute for Health Policy. Health information technology in the united states: The information base for progress. <http://www.policyarchive.org/handle/10207/bitstreams/21618.pdf>, February 2008.

- [16] Carl A. Gunter, David M. Liebovitz, and Bradley Malin. Experience-based access management: A life-cycle framework for identity and access management systems. *IEEE Security & Privacy Magazine*, 9(5), September/October 2011.
- [17] Health information technology for economic and clinical health act, subtitle d, § 13402.
- [18] HealthIT.gov. Benefits of ehrs: Health care quality & convenience. <http://www.healthit.gov/providers-professionals/health-care-quality-convenience>.
- [19] HealthIT.gov. Benefits of ehrs: Improved care coordination. <http://www.healthit.gov/providers-professionals/improved-care-coordination>.
- [20] HealthIT.gov. Benefits of ehrs: Improved diagnostics & patient outcomes. <http://www.healthit.gov/providers-professionals/improved-diagnostics-patient-outcomes>.
- [21] Eric Jamoom, Vaishali Patel, Jennifer King, and Michael Furukawa. National perceptions of ehr adoption: Barriers, impacts, and federal policies, August 2012.
- [22] Department of Health and Human Services. Notice of proposed rulemaking. *Federal Register*, 76(104), 2011.
- [23] United States Department of Health & Human Services. Hitech act enforcement interim final rule. <http://www.hhs.gov/ocr/privacy/hipaa/administrative/enforcementrule/hitechenforcementiffr.html>.
- [24] The Department of Health and Human Services. Summary of the hipaa privacy rule. <http://www.hhs.gov/ocr/privacy/hipaa/understanding/summary/index.html>.
- [25] American College of Physicians (ACP). Remark on nprm. http://www.acponline.org/advocacy/where_we_stand/health_information_technology/hipaa_disclosures.pdf.
- [26] American College of Surgeons (ACS). Remark on nprm. <http://www.facs.org/ahp/hipaa-acctletter.pdf>.
- [27] World Health Organization. International classification of diseases (icd). <http://www.who.int/classifications/icd/en/>.
- [28] World Wide Web Consortium (W3C). Extensible markup language (xml) 1.0. <http://www.w3.org/TR/xml/>.
- [29] Mark Weiner. Implications of the health insurance portability and accountability act of 1996. <http://www.cs.princeton.edu/courses/archive/spr02/cs495/HIPAA-princeton.pdf>, February 2002.
- [30] He Zhang, Sanjay Mehrotra, David Liebovitz, Bradley Malin, and Carl A. Gunter. A patient flow model for scoring anomalies in access logs. Technical report, Northwestern University, 2011.

Appendix A

Database Schema

A.1 Schema Diagram

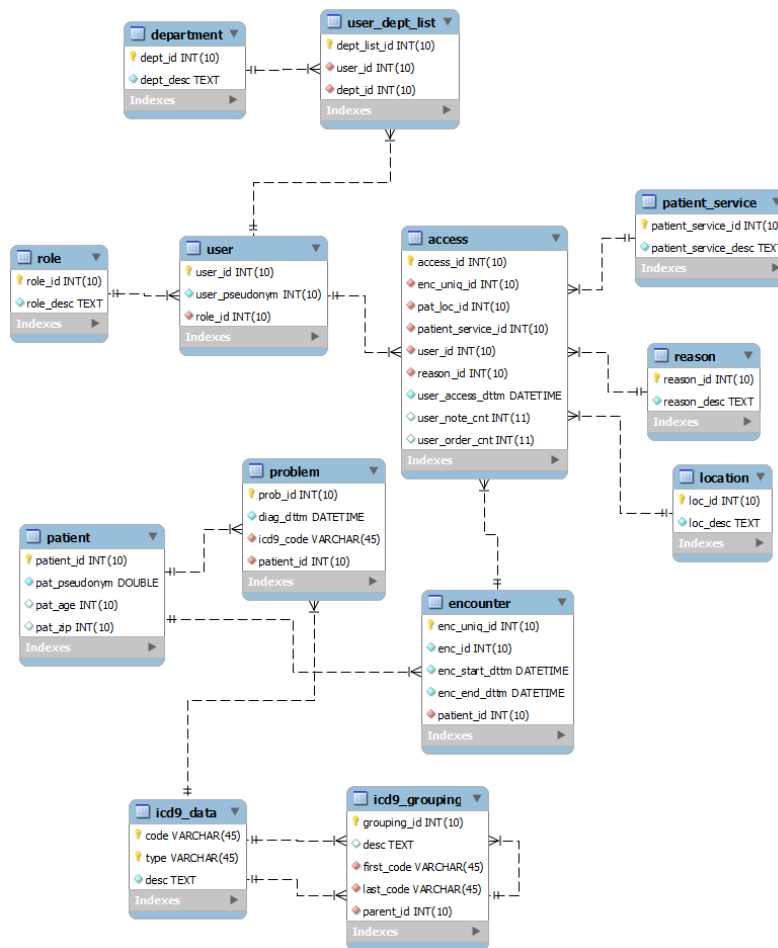


Figure A.1: Entity Relationship Diagram

A.2 Table Descriptions

access

Field	Type	Reference	Description
access_id	int(10) UN PK AI		Primary Key
enc_uniq_id	int(10) UN	encounter	Reference to the encounter of this access
pat_loc_id	int(10) UN	location	Reference to the location within the hospital
patient_service_id	int(10) UN	patient_service	Reference to the patient's service ID
user_id	int(10) UN	user	Reference to the user making the access
reason_id	int(10) UN	reason	Reference to the user-supplied reason
user_access_dttm	datetime		Date and time of access
user_note_cnt	int(11)		Number of notes accessed
user_order_cnt	int(11)		Number of orders accessed

department

Field	Type	Reference	Description
dept_id	int(10) UN PK AI		Primary Key
dept_desc	text		Name of the department

encounter

Each access belongs to an encounter, which is essentially a patient's time at the hospital from check-in until discharge.

Field	Type	Reference	Description
enc_uniq_id	int(10) UN PK AI		Primary Key
patient_id	int(10) UN	patient	Reference to the patient for this encounter.
enc_end_dttm	datetime		The end date and time of the encounter.
enc_id	int(10) UN		The encounter pseudonym generated by NMH
enc_start_dttm	datetime		The start date and time of the encounter.

icd9_data

This table encapsulates an ICD-9 code. The code and type together form the primary key.

Field	Type	Reference	Description
code	varchar(45) PK		The ICD-9 code itself
type	varchar(45) PK		Either diagnosis or procedure
desc	text		A textual description of the code

icd9_grouping

This table handles the hierarchical structure of the ICD-9 codes.

Field	Type	Reference	Description
parent_id	int(10) UN	icd9_grouping	The higher-level ICD-9 code.
grouping_id	int(10) UN PK AI		Primary Key
first_code	varchar(45)		For higher-level groups, the starting code
last_code	varchar(45)	icd9_data	For higher-level groups, the ending code
desc	text		

location

This table holds the locations within NMH.

Field	Type	Reference	Description
loc_desc	text		The name of the location
loc_id	int(10) UN PK AI		Primary Key

patient

This table encapsulates all the relevant patient information available in the de-identified data set.

Field	Type	Reference	Description
pat_zip	int(10) UN		ZIP code, de-identified to the first three digits.
pat_age	int(10)		The patient's age
pat_pseudonym	double UN		The original NMH patient pseudonym
patient_id	int(10) UN PK AI		Primary Key

patient_service

This table holds the services into which a patient can be placed. Examples include "Hematology" and "Transplant Surgery."

Field	Type	Reference	Description
patient_service_desc	text		The textual description, as above
patient_service_id	int(10) UN PK AI		Primary Key

problem

Field	Type	Reference	Description
diag_dttm	datetime		The date and time of diagnosis
prob_id	int(10) UN PK AI		Primary Key
icd9_code	varchar(45)	icd9_data	The ICD-9 code for this problem
patient_id	int(10) UN	patient	The patient affected by the problem

reason

A reason for access, usually closely resembling the user role.

Field	Type	Reference	Description
reason_id	int(10) UN PK AI		Primary Key
reason_desc	text		Textual description of the reason

role

Field	Type	Reference	Description
role_desc	text		Name of the role
role_id	int(10) UN PK AI		Primary Key

user

Field	Type	Reference	Description
user_pseudonym	int(10) UN		Original NMH user pseudonym
user_id	int(10) UN PK AI		Primary Key
role_id	int(10) UN	role	A reference to the user's role

user_dept_list

Maintains a list of departments for each user.

Field	Type	Reference	Description
dept_id	int(10) UN	department	The department linked to the user
user_id	int(10) UN	user	The user belonging to the department
dept_list_id	int(10) UN PK AI		Primary Key