

**THE USE OF SNOMED TO ENHANCE QUERYING  
OF A CLINICAL DATA WAREHOUSE**

by

Michael I. Lieberman, MD

A THESIS

Presented to the Division of Medical Informatics and Outcomes Research

and the Oregon Health & Sciences University

School of Medicine

in partial fulfillment of

the requirements for the degree of

Master of Science

May 2003

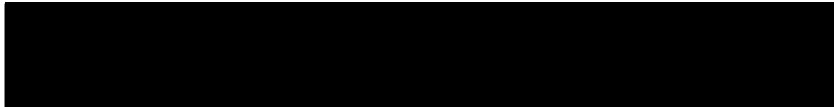
School of Medicine  
Oregon Health & Science University

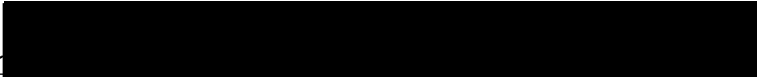
---

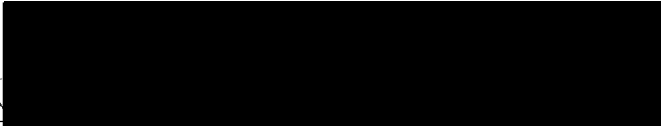
**Certificate of Approval**

---

This is to certify that the Master of Science thesis of  
Michael I. Lieberman  
has been approved

  
Kent A. Spackman, MD, PhD, Professor in charge of thesis

  
Judith R. Logan, MD, MS, Member

  
Thomas N. Ricciardi, PhD, Member

## Acknowledgements

I would first like to thank my thesis committee chairman Kent A. Spackman, MD, PhD and thesis committee members Judith R. Logan, MD, MS and Thomas N. Ricciardi, PhD for their assistance and guidance throughout the thesis process. Additionally, Thomas Ricciardi, PhD also provided invaluable assistance by creating the control queries used in my study and by instigating my interest in clinical data warehousing. I would also like to thank F.E. “Chip” Masarie, MD for his help and guidance in helping me to understand the finer points of medical terminology issues.

I would like to thank GE Medical Systems and the Medical Quality Improvement Consortium for making their clinical data warehouse available to me for this study.

Finally, I would like to thank my wife, Susan Hanson, and children, Ella and Joseph, for providing the inspiration, understanding, and loving support that made this work possible.

# TABLE OF CONTENTS

<b>ACKNOWLEDGEMENTS .....</b>	<b>I</b>
<b>ABSTRACT .....</b>	<b>IV</b>
INTRODUCTION .....	IV
METHODS .....	IV
RESULTS .....	V
CONCLUSION.....	VI
<b>INTRODUCTION.....</b>	<b>1</b>
PREVIOUS WORK .....	2
<i>Johns Hopkins</i> .....	3
<i>Medical Entities Dictionary</i> .....	3
<i>Partners Healthcare</i> .....	4
<i>TrialDB</i> .....	5
<i>Intermountain Healthcare (IHC) / 3M</i> .....	6
<i>Literature Summary</i> .....	6
<b>METHODOLOGY .....</b>	<b>7</b>
BACKGROUND.....	7
<i>EMR and User-interface Terminology</i> .....	7
<i>SNOMED CT</i> .....	11
MAPPING.....	13
<i>Problems</i> .....	13
<i>Medications</i> .....	16
PHASE I.....	17
<i>Research Hypothesis</i> .....	17
<i>Method</i> .....	18
PHASE II.....	22
<i>Research Hypothesis</i> .....	22
<i>Method</i> .....	23
<i>Measurements and Study Design</i> .....	25
<i>Subjects and Sample Size</i> .....	26
<b>RESULTS .....</b>	<b>26</b>
PHASE I.....	26
PHASE II.....	29
<b>DISCUSSION .....</b>	<b>36</b>
OVERVIEW .....	36
RELATIONSHIP TO PREVIOUS WORK .....	37
LIMITATIONS.....	38
<i>Phase I</i> .....	38
<i>Phase II</i> .....	40
<b>CONCLUSION .....</b>	<b>41</b>

<b>REFERENCES.....</b>	<b>42</b>
<b>APPENDIX 1: QUERY TOOL INSTRUCTIONS .....</b>	<b>44</b>
INTRODUCTION .....	45
PART 1, TERM LOOKUP.....	45
<i>SNOMED CLUE Browser.....</i>	<i>45</i>
<i>GE Terminology Concept Search .....</i>	<i>47</i>
PART 2, QUERIES .....	47
<i>Main Query Screen .....</i>	<i>48</i>
<i>Concept Description Screen .....</i>	<i>49</i>
<i>Concept Code Entering Screen.....</i>	<i>50</i>
<b>APPENDIX 2: SNOMED CONCEPT – GPI STEM MAPPING.....</b>	<b>51</b>
<b>APPENDIX 3: PHASE II CONSENT FORM.....</b>	<b>54</b>
<b>APPENDIX 4: QUERY TOOL APPLICATION CODE.....</b>	<b>58</b>

# **Abstract**

## ***Introduction***

With ongoing implementation of electronic medical records (EMR), more and more clinical data are being recorded in a digital format[1], often as free text[2]. Analyzing data in free text format is inefficient, prone to error, and difficult. While mapping free text to coded concepts can address some of these issues, additional structure is necessary to enable non-expert end-users to retrieve data from a clinical warehouse. A review of the current literature reveals that this structure is often built and maintained locally with significant effort[3]. Alternatively, one can use an external reference terminology to provide the necessary structure. This study examined whether one such reference terminology, SNOMED Clinical Terms® (SNOMED® International, Northfield, IL), could enhance the querying of a clinical data warehouse.

## ***Methods***

The study was designed to test the following two hypotheses:

- 1. Using a hierarchically-organized concept-based reference terminology will simplify the creation and maintenance of queries without significant degradation of query results.*
- 2. When non-experts formulate queries, the use of a hierarchically-organized concept-based reference terminology will improve query results and the time taken to perform queries.*

In the first of two phases of this study, the level of agreement in result sets obtained using a SNOMED-based query versus a query developed by analysts without using SNOMED was examined. In the second phase, twelve test subjects were used to compare query results using both a SNOMED-based tool and a non-hierarchical terminology based tool. Recall, precision, and time taken to complete each of two queries were measured for each subject using each tool.

## **Results**

In Phase I, it was shown by inspection that a SNOMED-based query was easier to construct and maintain than an ad-hoc reference query created by MQIC analysts from the Medical Quality Improvement Consortium (MQIC) (GE Medical Systems Information Technology, Waukesha, WI). When the SNOMED-based query was compared to the MQIC reference query, the recall and precision for seventeen concepts were uniformly high, with recall ranging from 0.74 to 1.00 and fifteen of the concepts having a recall greater than 0.92, and precision ranging from 0.98 to 1.00. In Phase II, twelve subjects had a significantly higher recall and shorter query time for queries using the SNOMED-based method. The mean recalls were 0.99 and 0.65 using a SNOMED-based query tool for the two different queries compared to 0.40 and 0.37 using the non-hierarchical system. The mean times it took to complete each query were 368 seconds and 302 seconds for the SNOMED-based query versus 761 seconds and 595 seconds for the non-hierarchical system. There was no significant difference in precision between the two methods for either query.

## ***Conclusion***

SNOMED, a hierarchically-organized concept-based reference terminology, simplifies the creation and maintenance of clinical queries without significant degradation of results when compared to ad-hoc queries created by clinical data warehouse experts. In addition, a SNOMED-based query tool provided an improvement in recall and time taken to complete queries compared to a tool based on a non-hierarchical vocabulary in queries performed by non-experts.



## Introduction

With ongoing implementation of electronic medical records (EMR), more and more clinical data are being recorded in a digital format[1]. While some benefits of an EMR, such as universal access, can be realized regardless of how the data are stored, the true power of an EMR cannot be unleashed without providing some structure to the stored data. Structured data is required for many, if not all, of the processes that can improve quality and safety in the healthcare system, such as population-based management of high-cost preventable diseases, decision support using evidence-based guidelines, outcomes research, measurement of healthcare delivery system performance, and the conducting and evaluation of clinical trials. Unfortunately, more structure has historically meant more difficulty in entering data[2], and much information is captured as free text rather than coded data.

Although it is possible to analyze data in free text format, it is inefficient, prone to error, and difficult. For example, if one wants to identify patients with hypertension by searching problem list entries, one would have to do full text searches for partial matches (inefficient), which might include patients with a *family history of* hypertension or *ocular* hypertension (prone to error), and one would have to search for many different phrases that mean hypertension (difficult). Storing data as concepts represented by codes rather than free text strings will solve the problems of inefficiency and errors of inappropriate inclusion. Using concepts alone will also help with, but not solve, the third issue of having to search for multiple strings by having a concept encompass all synonyms. To adequately address this third issue, a search regarding a certain concept should also return

more specific concepts. For example, one would want patients with a problem of *malignant hypertension* when searching for *hypertension*.

The overall system requires additional knowledge to aggregate terms that represent a concept. This knowledge can be applied in at least three different ways. First, an expert in the local terminology can use his or her knowledge to create the appropriate query. Second, a layer of custom meta-data can be developed for an individual system that defines hierarchical relationships between clinical concepts and can be applied to the terminology. This meta-data can range in complexity anywhere from lists of concepts related to a single concept to a complex hierarchical vocabulary. Third, one can apply an external reference terminology, such as SNOMED Clinical Terms® (SNOMED CT®) (SNOMED® International, Northfield, IL), and use its hierarchical relationships to facilitate aggregation. The purpose of this research project was to pursue this third option and measure how well SNOMED CT, a hierarchically-organized concept-based reference terminology, facilitates data retrieval.

## ***Previous Work***

While there has been significant reporting on both medical vocabularies and data warehousing, no published studies attempted to vigorously measure the cost and benefits of using structured vocabularies to aid in the extraction of information from CDWs. One possible explanation is that the benefits are self-evident and do not need to be measured. Indeed, there were frequent assertions in the literature that controlled medical vocabularies are necessary to realize the full benefit of medical informatics

applications[4-6]. Forman, et al. write, “Applying an organizing framework to data transforms it into useful information which is analyzable and comparable.”[7]. Despite such assertions, the use of controlled medical vocabularies is not ubiquitous. One important reason for this is that a clear business case cannot always be made to support adoption of a comprehensive terminology[8]. Therefore, it does not appear that the benefits of terminologies are self-evident. While no studies measuring the impact of a structured vocabulary on querying CDWs were found, the literature does report on various experiences extracting information from CDWs.

## **Johns Hopkins**

Stoffel, et al. describe the development of a graphical ad-hoc query tool[9]. The impetus for this project was the realization that common tools used in other domains such as online analytical processing (OLAP) and materialized views did not port well to medical databases. A major goal of this project was to create a data structure in the form of a directed acyclic graph (DAG) to facilitate hierarchical searching. Evaluation of their query tool was done by comparing ease of use in generating queries using the tool compared to generating queries without it. The only data given pertaining to this measure was that a tool-based query required the selection of only two attributes, while a query without the tool would require including over 300 values[9].

## **Medical Entities Dictionary**

The Medical Entities Dictionary (MED) is a robust controlled medical terminology that has been developed and is in use at Columbia Presbyterian Medical Center (CPMC). The

MED has allowed CPMC to integrate data from a myriad of systems in the clinical data repository and implement a variety of applications, such as decision support and data mining[7, 10]. However, development, implementation, and maintenance of this vocabulary has taken, and continues to take, a great deal of effort[3]. In 1995, the MED was averaging 4.55 updates each month, with a typical update containing 83 new concepts[7]. The MED has been implemented with an Object Oriented Database structure, which has been reported to facilitate maintenance[11].

## Partners Healthcare

Partners Healthcare maintains a relational database (separate from the production database used to store patient information) that it uses to select patients for possible participation in clinical trials. In designing this database, they reviewed 16 years of queries of the clinical database[12]. They found that 78.8 % of queries could be answered using only diagnoses, medications, and procedures. Adding all types of lab values would cover 92.5% of queries and the addition of clinical text would cover all queries. They eventually created a star schema with two fact tables and six dimension tables. They also realized that most queries were “based upon hierarchical lists of diagnosis and medication codes”.[12] For this reason, they included a *search hierarchy* attribute in their concept dimension. A hierarchical structure is maintained in their vocabulary by assigning DOS file structure-like strings to the *search\_hierarchy* attribute so that concept sub-types can be easily retrieved through SQL constructs using the *like* operator[13]. Maintenance of the concept table, including the hierarchical relationships, requires approximately one full-time analyst position[13].

## TrialDB

The TrialDB (formerly ACT/DB) system has been developed as a clinical study data management system (CSDMS)[14]. This system uses entity-attribute-value (EAV) tables to store most clinical data. The developers of this system have created meta-data layers and user interface tools to provide users of the system with the feel of a conventional system with one large table where each attribute is represented by one field in the table. This is done because, through use of spreadsheets and statistical programs, end users expect data to be presented in such a manner[15]. A key component of their system is its ad-hoc query tool. Early on, they realized that creating pre-defined queries for use by end users would neither be sufficient, as the variability of queries would be too high, nor maintainable, as the queries would need to be updated as the database evolved[16].

Although the developers have created a sophisticated query tool, as well as a data dictionary, they have not implemented a hierarchical structure to their data, though this task is listed as an area of future work[16]. One possible explanation for not yet implementing a hierarchy is two-fold. First, the vast majority of attributes stored in their database are either lab tests or clinical findings[14], for which a hierarchical structure might not be as important as for problems or medications. Second, the database stores information about clinical studies rather than a general clinical practice, and queries may more often involve findings rather than problems and medications.

## **Intermountain Healthcare (IHC) / 3M**

IHC has developed the Enterprise Data Warehouse (EDW), which uses the internally-developed 3M Healthcare Data Dictionary (HDD)[17]. The HDD uses pre-existing coding classifications and vocabularies such as SNOMED and LOINC as “starter sets” or reference sources whenever possible[18]. The structure of the HDD has been used to facilitate the mapping of laboratory result terms from multiple sources to LOINC[18]. The EDW generates an average of 85,000 queries a month. The HDD has been noted to provide a significant benefit, both through translating terms to common concepts and by providing relationships between concepts[17].

### **Literature Summary**

There is widespread agreement that extensive use of reference terminologies is necessary to realize the full benefits of informatics applications. However, the bulk of the literature is descriptive rather than analytic. Most, but not all, clinical data warehouses reported in the literature utilize terminologies with hierarchical relationships. All the reports have described locally-developed vocabularies that have been successful, though resource intensive.

# **Methodology**

## ***Background***

### **EMR and User-interface Terminology**

Logician® (GE Medical Systems Information Technology (GEMS), Waukesha, WI) is a leading ambulatory electronic medical record (EMR). Logician has a target market of medium to large healthcare institutions throughout the United States. Logician allows users quite a bit of latitude in data entry. While some fields can be populated by predefined code sets, free text can also be used in many, if not most, fields. Patient information is stored in a variety of tables in an Oracle® (Oracle Corporation, Redwood Shores, CA) relational database at each customer site.

### ***GEMS Terminology***

The GEMS terminology was created as a user-interface terminology as opposed to a reference terminology, and therefore, there has been no attempt to create relationships between the concepts[8]. The GEMS terminology began in the area of observations. In conjunction with its customers, GEMS assigned unique codes for various observations that its customers wanted to report. In order to maintain consistency across its customers, additions to the terminology had to go through GEMS to ensure that there were no duplicates in meaning or name[8].

In order to provide reporting, and eventually decision support, for users of this system, GEMS began mapping strings that were being entered by the system's users to specific concepts. Starting with the most frequently used strings, the mapper would investigate whether a particular string had the same meaning as an already present concept, in which case the particular string would be added to the terminology as an *entry term* for that concept. If the string did not map to a present concept, a new concept would be created. A user might also have entered an ICD-9 code along with the string. Because GEMS found that at times the ICD-9 code did not correlate with the problem string, this code was not used in the mapping process.

### *Medical Quality Improvement Consortium*

In order to better utilize the vast amounts of clinical data being collected, GEMS formed the Medical Quality Improvement Consortium (MQIC) in 2001 to pool de-identified data from its customers. The stated objectives of MQIC include providing access to anonymous, patient de-identified pooled data for research and quality improvement purposes, as well as providing summary reports and information about patterns of care for clinics, providers, and patients[19]. The MQIC database currently contains information on over half a million patients from ten different sites. The architecture of MQIC (Figure 1) involves extracting data at each participating site and importing the de-identified data to the central staging area. The information is then processed into a set of fact and dimension tables creating a star schema in the MQIC clinical data warehouse (CDW) (Figure 2).



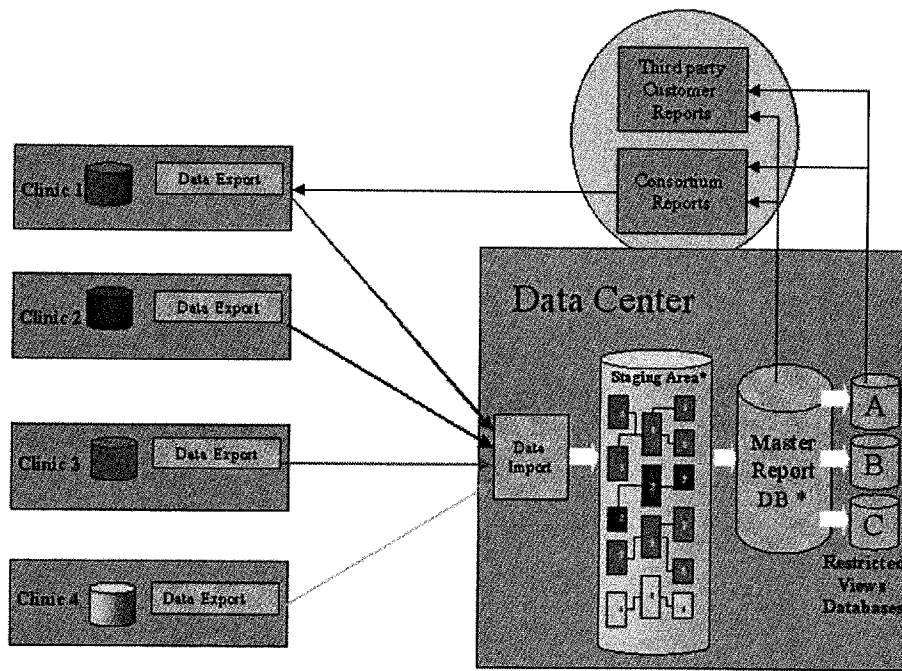


Figure 1. MQIC Architecture

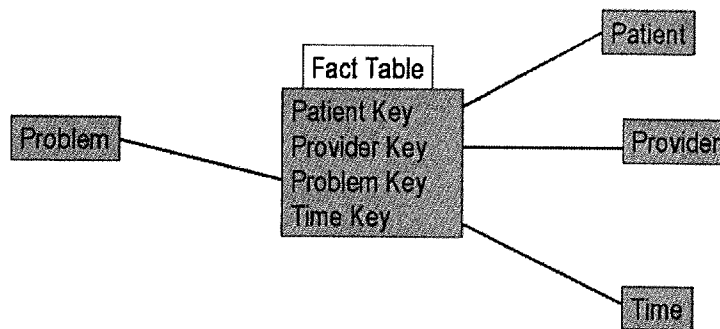


Figure 2. MQIC Dimensional Model

There are fact tables for problems, medications, and observations. Each fact table relates pointers to various dimension tables. For example, a row of the problem fact table would include a problem concept key, a patient key (that is different from the patient id used at the originating site), a date key, and a provider key. The information that expands each

key is held in separate dimension tables. Table 1 shows an example of a row from the problem dimension table:

<b>Problem_key</b>	<b>Problem_string</b>	<b>Concept_id</b>	<b>Concept_description</b>	<b>ICD-9</b>
7567843	HTN	281	<i>hypertension</i>	401

**Table 1. Problem Dimension Example**

The problem key serves as the primary key that links this dimension table to the fact table. The problem string is the actual string imported from the clinical system. If this string has previously been mapped to a GEMS concept, the concept id and concept description fields will be populated. Additional information pertinent to the concept, such as ICD-9 code, may also be present. For example, if the string *myocardial infarction* is imported as a problem, the problem dimension table will be searched to see if it already has an entry for this string. If a key already exists, this problem key will be entered in the problem fact table. If this string had previously been mapped to a concept, a concept code will already exist in the dimension table, and this new fact will then be associated with that concept. It is possible that a problem key will not have a concept code if either the problem string has never been seen before, in which case a new problem key would be created, or the problem string has not yet been mapped to a GEMS concept.

In general, MQIC analysts do all of the CDW querying. The query process begins with a plain English request going to an analyst with a thorough knowledge of the MQIC data. After refining the query request to reflect what data is available, it is passed onto a database specialist who creates the SQL and runs the query. The problem of aggregating



concepts is addressed by keeping sets of concepts that should be included within a diagnosis and using these sets in creating queries. In addition to GEMS concepts, the analysts can also use ICD9-CM codes, as well as partial string matches to enhance retrieval. Because a percentage of problem strings have not been mapped to GEMS concepts (approximately 1% - 20% depending on the clinical area), recall can be compromised. Partial string matches, by accessing all entries in the fact table, can be quite effective in increasing recall. However using this method as an overall search strategy can decrease precision by introducing inaccuracies as mentioned in the introduction.

## **SNOMED CT**

SNOMED CT is a comprehensive healthcare terminology that was created by combining SNOMED RT and Read Codes Version 3[20]. SNOMED began over thirty years ago as the Systemized Nomenclature of Pathology. Over the years, it has been expanded and revised so it is now a true ontology with multiple hierarchies and description logic based relationships between its concepts[21]. The primary hierarchy is based on the *Is-A* relationship relating every concept to one or more super-types.<sup>1</sup>

Operationally, the core of SNOMED is contained in three tables: Concepts, Descriptions, and Relationships (Figure 3)[22]. There is a single entry for each unique concept in the Concept table that lists the concept code, status of the code (e.g. current, retired, outdated), fully specified name of the concept, and legacy codes. Entries in the

---

<sup>1</sup> The root concept, *SNOMED CT Concept* has no super-type.

Descriptions table relate specific terms to associated concepts. Each entry in the Relationships table lists a source concept, a relationship type (such as *Is-A* or *Has-Active-Ingredient*), and a target concept.

Through the use of description logic, it is possible to express many concepts as combinations of other concepts and relationships, which is known as post-coordination. For example, *myocardial infarction* could be expressed as a *disease* that has finding site *myocardial structure* and associated morphology *infarct*. Though semantically correct, post-coordination is difficult from a clinical perspective. Therefore, a great many concepts, such as *myocardial infarction*, are pre-coordinated.

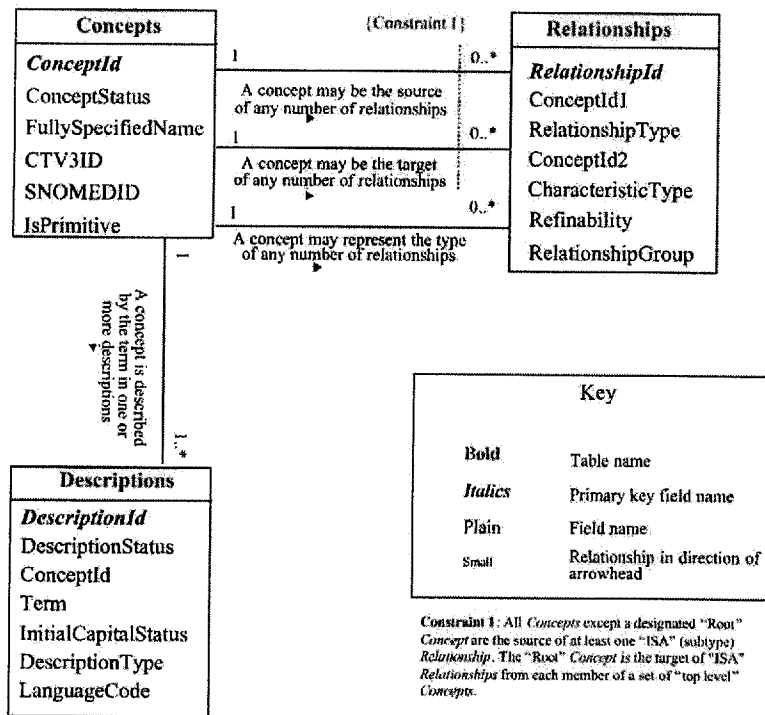


Figure 3. The SNOMED Clinical Terms Core Structure

SNOMED was chosen as the reference terminology for this study because multiple studies comparing it to other coding schemes and vocabularies have found it to be the most complete in coverage of medical concepts[23-26].

The primary goal of this study was to investigate the effectiveness of a hierarchical reference terminology, SNOMED, in facilitating data retrieval from a clinical data warehouse. In order to do this, it was necessary to map GEMS concepts to SNOMED. Ideally, the research would have been best carried out by mapping the entire GEMS terminology to SNOMED. However, mapping is non-trivial and time-consuming. One study noted that it took 350 hours to encode 2,379 codes[26]. For this reason, the study was confined to a limited clinical space: cardiovascular disease.

## ***Mapping***

In order to investigate the ability of SNOMED to improve CDW querying, GEMS concepts in the domain of cardiovascular disease needed to be mapped to SNOMED. Specifically, concepts related to heart failure (HF), coronary artery disease (CAD), hypertension, atrial fibrillation, and the medications used to treat these disorders were mapped. Because the presence or absence of diabetes mellitus (DM) often affects the course and management of these diseases, concepts related to it were also mapped. The process of mapping problems and medications differed.

## **Problems**

Initially, the GE vocabulary was searched for all problem concepts related to the target disorders mentioned above. Next, the most appropriate SNOMED concept or concepts

for each GE concept was found using the SNOMED CLUE Browser (The Clinical Information Consultancy, Reading, Berkshire, Great Britain). Overall, 271 problem concepts were mapped. Table 2 displays how well these concepts were mapped.

Quality Code	Meaning	GE Codes
1	One to One Map	126
2	SNOMED more specific	5
3	GE more specific	85
4	Ignores s/p or h/o	11
5	Similar	11
6	Compound concept	31
7	Unmappable	2

**Table 2. Quality of Mapping**

Almost half of the GE concepts had a direct one-to-one match in SNOMED, which led to mappings with Quality Code 1. The next most common type of mapping involved GE concepts needing to be mapped to a more general SNOMED concept, Quality Code 3. An example of this is the GE concept *mild hypercholesterolemia* that was mapped to the SNOMED concept *hypercholesterolemia*. This type of mapping allowed the more specific GE concept to be retrieved in a query using the more general SNOMED concept. However, without using post-coordination within SNOMED to assign the qualifier *mild* to this concept, some of the information of the GE concept was lost.

Compound concepts, Quality Code 6, were the next most common mapping category, with 31 instances. Because the GE terminology is used as an interface terminology, it allows compound concepts such as *hypertension, with left ventricular hypertrophy*. This GE concept was mapped to the two different SNOMED concepts, *hypertensive disorder*

and *left ventricular hypertrophy*. This type of mapping was actually a form of post-coordination, and no information was lost.

There were eleven instances each for the categories of similar concepts, Quality Code 5, and concepts that ignore *status post (s/p)* or *history of (h/o)* phrases, Quality Code 4. For similar concepts (Quality Code 5) such as the GE concept *diabetes mellitus, with neurological complications* and the SNOMED concept *diabetes mellitus with neurological manifestation*, there was more value in linking the two than moving to a more general SNOMED concept, such as *diabetes mellitus*, where too much information would be lost. For GE concepts that were of the type *h/o* or *s/p* where there was no one-to-one SNOMED match, the *h/o* or *s/p* was ignored when mapping to a SNOMED code (Quality Code 4). While there are instances when this type of qualifier may be necessary for correct retrieval (e.g. finding information related to a hospitalization where a specific procedure was done), in most cases the qualifier does not alter the meaning of the concept from the perspective of someone selecting patients from the database. For example, when looking for diabetics, one would want to retrieve patients with the problem of *h/o diabetes mellitus*.

Five of the GE concepts were mapped to more specific SNOMED concepts, Quality Code 2. An example of this occurrence is the GE concept *h/o angina* that was mapped to the SNOMED concept *h/o angina pectoris*. While, in general, this sort of mapping should be avoided for fear of retrieving inappropriate results, in each case it seemed unlikely that an analyst would truly want to exclude the less specific GE concept when a query with the more specific SNOMED concept was run.



Only two concepts were unmappable and they were both of the *rule out (r/o)* construct. Though SNOMED does contain a concept for *rule out*, one cannot simply use the compound mapping technique because it does not allow for linkage between the two concepts. With compound mapping, if a patient had only a *r/o* concept listed, such as *r/o diabetes mellitus*, he or she would be retrieved with a query for the second part of the concept, such as *diabetes mellitus*, which would be inappropriate in that the diagnosis was only considered and not made.

## Medications

Because the GE vocabulary has separate concepts for each brand name and formulation of a medication, there were 1,039 GE concepts in the chosen domain that needed to be mapped. Due to combination medications, such as *triamterene/hydrochlorothiazide*, that required mapping to multiple SNOMED concepts, there were 1,220 mappings required. In order to simplify this process, instead of manually mapping all of these concepts, the generic product index (GPI), which had been assigned to each GE concept, was used to map that concept to a SNOMED medication class, such as *thiazide diurectic* or *potassium-sparing diuretic* in the above example.

The GPI is a 14-digit code where the first ten digits specify the medication and the last four digits specify the dosage and type of formulation. The left-most digits specify the medication class, with the level of detail increasing with further digits. For example, beta-blockers all start with 33, while beta-1 specific beta-blockers start with 332. Through the mapping of 44 GPI stems (e.g. 33 for beta-blockers) to one or more

SNOMED medication classes, 1,220 concept mappings were accomplished. It was necessary to map a single GPI stem to more than one SNOMED concept when that GPI stem represented a combination medication. These compound mappings assured that when a query is done for a particular medication, patients who are on a combination including that medication would be retrieved. A table of the GPI Stem – SNOMED Medication Class mappings is available in Appendix 2.

To investigate the usefulness of a hierarchically-organized concept-based reference terminology to enhance querying of a CDW, the study was divided into two phases.

## ***Phase I***

### **Research Hypothesis**

The hypothesis for the Phase I of the study was:

*Using a hierarchically-organized concept-based reference terminology will simplify the creation and maintenance of queries without significant degradation of query results.*

Evaluation of this hypothesis involved two steps. First, queries created using SNOMED were compared by inspection to queries created in an ad-hoc manner without the use of SNOMED. Next, the results of these two queries were compared to evaluate whether the SNOMED-based query performed as well as the non-SNOMED-based query.

## Method

MQIC keeps copies of all the query requests that they perform on the CDW. The set of concepts from these queries that addressed the clinical domains where mapping had been done were selected. The reference sets of de-identified patient ids were created by running queries developed by GE analysts for each concept. Because the SNOMED queries could only retrieve information that had been mapped to GEMS concepts, the queries were performed on the subset of MQIC patients with fact table data that had been mapped to GEMS concepts to ensure that each query was accessing the same patient base. The patient sets associated with each concept derived from the SNOMED-based query were created following the steps outlined in Table 3.

1. Select the appropriate SNOMED concept.
2. Select all sub-type concepts in the hierarchy using the Oracle SQL *CONNECT BY ... PRIOR* construct.
3. Select all the GEMS concepts that map to the selected SNOMED concept.
4. Run the query.

**Table 3. SNOMED-based Query Process**

Because results from the GE query were to be used as a reference set, they needed to be as accurate as possible. Therefore, after comparing initial results from the two queries, the GE query was refined to improve its accuracy when the SNOMED query had discovered a significant number of patients related to concepts that should have been included in the reference set.

A sample SNOMED query for coronary artery disease follows in Table 4:

```

select distinct patient_key
from problem_fact f, problem_dim d
where f.problem_key = d.problem_key and
d.concept_id in (
    select gems_concept_id from map_table
    where sct_concept = 8957000 or
    sct_concept in (
        select concept1 from relationships
        connect by concept2 = prior concept1 and relationshiptype = 116680003
        start with concept2 = 8957000 and relationshiptype = 116680003
    )
)

```

**Table 4. Sample SNOMED Query**

Even with the two sub-queries, this query appears simpler than one that must aggregate and exclude codes and strings within the query. A sample MQIC query to return patients with coronary artery disease follows in Table 5:

```

select distinct f.patient_key
from mqic_problem_ff, mqic_problem_dd
where f.problem_key = d.problem_key and
(
  d.concept_desc like 'coronary artery%' or
  d.concept_desc like 'coronary athero%' or
  d.concept_desc like 'myocardial%' or
  d.concept_desc like 'angina pectoris%' or
  d.icd9_code like '414%' or
  d.icd9_code like '413.9%' or
  d.icd9_code like '410%' or
  d.icd9_code like '412%'
) and
(
  d.problem_string not like '%fh %' or
  d.problem_string not like 'fhx%' or
  d.problem_string not like 'hx, family%' or
  d.problem_string not like 'fam%' or
  d.problem_string not like 'questionable h/o%' or
  d.problem_string not like '%r/o%' or
  d.problem_string not like '%rule%' or
  d.problem_string not like 'risk of%' or
  d.problem_string not like '%family%' or
  d.problem_string not like '%father%' or
  d.problem_string not like '%mother%' or
  d.problem_string not like '?%' or
  d.problem_string not like '%question%' or
  d.problem_string not like 'hx of?%'
)

```

**Table 5. Sample MQIC Query**

While the logic of the MQIC query is simpler, it requires more effort and knowledge of the data to create. Furthermore, as the GEMS terminology expands, this query would need to be updated to incorporate these changes (e.g. if the concept *coronary steal syndrome* is added to the terminology, the string for this concept would need to be added to the query). With the SNOMED query, as long as new GEMS terminology concepts are mapped to SNOMED, the query does not have to be changed and will still return data related to the new concept. In addition, to query for a different concept using the

SNOMED construct, one has to change only a single value (the code for the concept of interest) rather than substitute sets of concept codes and partial string values, as would be necessary with the GEMS construct.

Therefore, by inspection, the SNOMED-based query is easier to construct and maintain. The next part of this phase involved investigation of whether using the SNOMED-based query caused a degradation in the results compared to the reference MQIC query. In terms of Table 5, a good result would be small values of *b* and *c* compared to *a*.

Retrieved by GE	Retrieved by SNOMED	
	Yes	No
Yes	a	b
No	c	d

**Table 6. Stage I Results Example**

Finding a test of statistical significance to definitively answer this question proved to be difficult. McNemar's test was considered first. At first glance, it appeared appropriate in that it is used to test paired binary data. However, it is truly a test of symmetry and measures only the relative sizes of *b* and *c* compared to each other<sup>2</sup> and does not take into account the positive concordant pairs (*a*). For this reason, it is not an appropriate statistic for this study. Next the Kappa statistic, which is used to measure inter-observer variability, was considered. Because it measures the size of concordant pairs in relation to discordant pairs, it, too, seemed to be an appropriate statistic. However, even for mediocre results, kappa was very high (>0.99) due to *d* being very large in relation to *a*,

---

<sup>2</sup> The actual statistic is  $(b-c)^2/(b+c)$ .

$b$ , and  $c$  because of the large set of patients being queried.<sup>3</sup> One could try to compensate for this by specifying a kappa above a certain value as significant. Yet determining that value would be arbitrary and based on what were acceptable values for  $a$ ,  $b$ , and  $c$ .

Instead, precision ( $a / (a + c)$ ) and recall ( $a / (a + b)$ ) were chosen to evaluate the SNOMED result in relation to the GE result. Although precision and recall do not directly provide for a test of statistical significance, they are common measures for the quality of a retrieval algorithm. In the framework of a clinical database, the SNOMED query method is such an algorithm.

## ***Phase II***

### **Research Hypothesis**

The hypothesis for the Phase II of the study was:

*When non-experts formulate queries, the use of a hierarchically-organized concept-based reference terminology will improve query results and the time taken to perform queries.*

The investigation of this hypothesis involved asking subjects to perform queries using two different methods and comparing the results.

---

<sup>3</sup> There are approximately 500,000 patients with problems or medications mapped to GEMS concepts.

## Method

This phase built on the results of Phase I. While Phase I addressed how SNOMED facilitated creation and maintenance of queries by experts, Phase II investigated whether the use of SNOMED could improve the quality of queries by non-experts. The user interface of the two tools was meant to be as similar as possible to make sure that the effect of the vocabulary was measured rather than the difference in user interface.

An example of the user interface can be seen in Appendix 1, Figure 3 through Appendix 1, Figure 5. Using an external browser for either SNOMED (Appendix 1, Figure 1) or the GEMS terminology (Appendix 1, Figure 2), the test subject cut and pasted concept codes into the query tool to form sets of concepts on which to search. The subject then added the set to the query as either attributes that the patients should or should not possess. When all of the various attributes had been entered, the subject could run the query. The full instructions that were given to the participants can be found in Appendix 1. In addition to the written instructions, each participant was given a demonstration of the system before beginning the study. Table 7 illustrates a brief example of how a subject would create a query using the SNOMED system to find all patients with *coronary artery disease* not on *aspirin*.



1. Begin the query by pressing **Add New Concept**.
2. Define the concept by assigning a label and specifying whether the concept is a medication or problem and whether patients with or without the concept should be found.
3. Use the SNOMED browser to find the concept code for *coronary artery disease* (8957000).
4. Paste this code into the concept code entering screen and press **Concept Complete**. If using the non-hierarchical system, one would need to keep adding concepts that fell under *coronary artery disease*.
5. Repeat steps 1 through 4 for the concept *Aspirin* (7947003).
6. Press **Query Complete**.

**Table 7. Query Creation Steps**

The time each participant took to complete each query using each system was also measured. To determine whether the vocabulary lookup tool had a significant effect on this time, each participant was also asked to look up two concepts using each system, and the times were recorded. Finally, each participant was asked to rate the ease of use and likelihood of reuse of each system in an on-line questionnaire (Figure 4).

Study 1:100 Questionnaire

For the following questions '5' is the best and '1' is the worst

<p>1. Please rate ease of use of the SNOMED interface (1 - very difficult, 5 - very easy)</p> <p style="text-align: center;"> <input type="radio"/> 1    <input type="radio"/> 2    <input type="radio"/> 3    <input type="radio"/> 4    <input type="radio"/> 5         </p>
<p>2. Please rate ease of use of the GEMIS method (1 - very difficult, 5 - very easy)</p> <p style="text-align: center;"> <input type="radio"/> 1    <input type="radio"/> 2    <input type="radio"/> 3    <input type="radio"/> 4    <input type="radio"/> 5         </p>
<p>3. How likely would you be to use the SNOMED method (1 - very unlikely, 5 - very likely)</p> <p style="text-align: center;"> <input type="radio"/> 1    <input type="radio"/> 2    <input type="radio"/> 3    <input type="radio"/> 4    <input type="radio"/> 5         </p>
<p>4. How likely would you be to use the GEMIS method (1 - very unlikely, 5 - very likely)</p> <p style="text-align: center;"> <input type="radio"/> 1    <input type="radio"/> 2    <input type="radio"/> 3    <input type="radio"/> 4    <input type="radio"/> 5         </p>

Figure 4. Study Questionnaire

## Measurements and Study Design

The recall, precision, and elapsed time were measured for each query using both vocabularies. The results of the MQIC query from Phase I served as the reference set. The study was a crossover design with half the subjects first using the SNOMED-based tool to perform the query set, then using the non-hierarchical system to perform the same queries. The other half of the subjects used the tools in the opposite order. Because the results of each query from each subject were being compared between the two tools, the paired Student's t-test was used to measure differences in the recall, precision, and time between the tools.

## Subjects and Sample Size

Subjects were masters-level graduate students enrolled in the Medical Informatics curriculum in the Division of Medical Informatics and Outcomes Research at Oregon Health and Sciences University (Portland, OR) during the Winter Term of 2003. This group of students was chosen because of their exposure to clinical medicine and computer science as part of the required Informatics curriculum. This program included both clinicians (i.e. physicians, nurses, pharmacists) and non-clinicians. A sample size of 9 subjects was calculated to detect a 0.2 difference in precision or recall with an alpha of 0.05 and a power of 0.9. A total of 12 subjects, 5 clinicians and 7 non-clinicians, participated in the study.

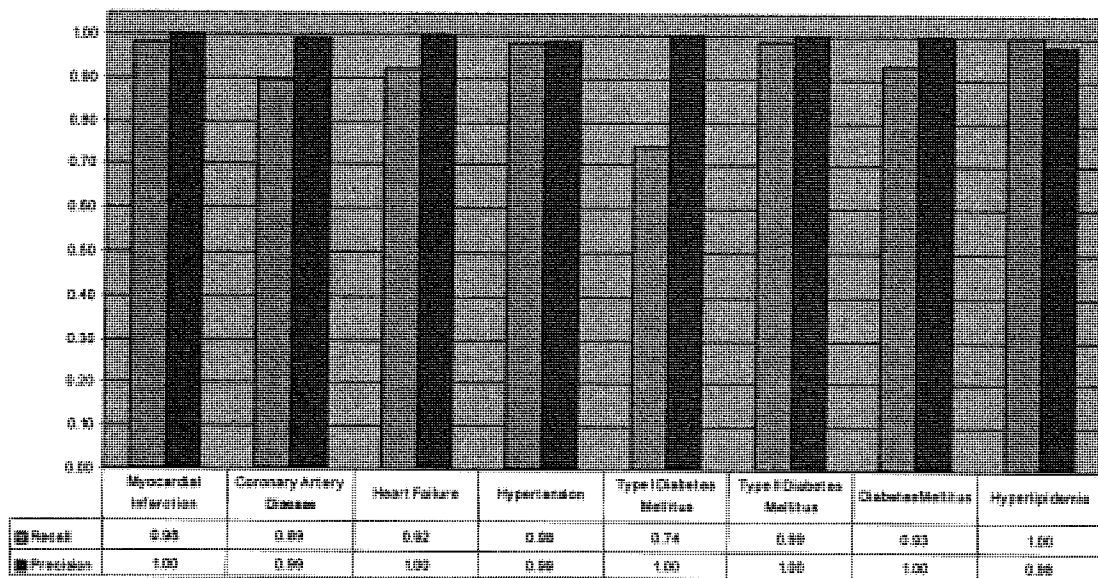
## Results

### *Phase I*

The concepts studied, as well as the precision and recall for each concept, are shown in Figure 5 and Figure 6. The sizes for retrieval sets of unique patient ids ranged from 1,489 for *Niacin* to 99,559 for *Hypertension*. The mean set size was 37,103 and the median was 27,899.

In general, precision was excellent, with a score of 1.00 for most concepts and no score lower than 0.98. In all cases except *HMGCoA Reductase Inhibitor*, a recall below 1.00 was caused by failure of the GEMS query to retrieve desired patients. For example, the reference query for *hypertension* used a combination of ICD-9 codes and the partial

string beginning with *hypertension*. This strategy failed to find concepts such as *renovascular hypertension* because the string did not start with *hypertension*. For *HMGC $\alpha$  reductase inhibitor*, a mistake in mapping caused patients on long-acting niacin to also be retrieved. Other than this mapping error, no occurrences where the SNOMED query actually retrieved any inappropriate patients were found. Because there were no truly false positives, the precision was, in effect, 1.00.



**Figure 5. Recall and Precision for PROBLEM Concepts**

Recall was also high, with the scores for all medications greater than 0.94 and all problems greater than 0.92, with the exception of *coronary artery disease (CAD)* and *type I diabetes mellitus (DM I)*. The slightly lower score of 0.89 for *CAD* was due to both a failure to map the GEMS vocabulary concept *atherosclerotic heart disease* to SNOMED and the MQIC query strategy that included subjects who had had a procedure that indicated the presence of *CAD*, such as coronary artery bypass graft (CABG). These

procedures were not directly specified in the query, but were rather picked up by searching on the string *coronary artery*.

The lower score for *DMI* was completely due to the inclusion of the GEMS concept *insulin dependent diabetes mellitus* in the reference set. The corresponding SNOMED concept is not a part of the *DMI* hierarchy. This concept was also not explicitly specified in the GEMS query, but was included through its association with ICD-9 codes that do specify *DMI*. This association occurred because a Logician user can enter both a text string and an ICD-9 code to describe a problem. The text string is what is mapped to a GEMS concept without consideration of the ICD-9 code. Therefore, the appropriateness of the retrieval depends on whether the original text string or ICD-9 code was more accurate and precise in recording the problem.

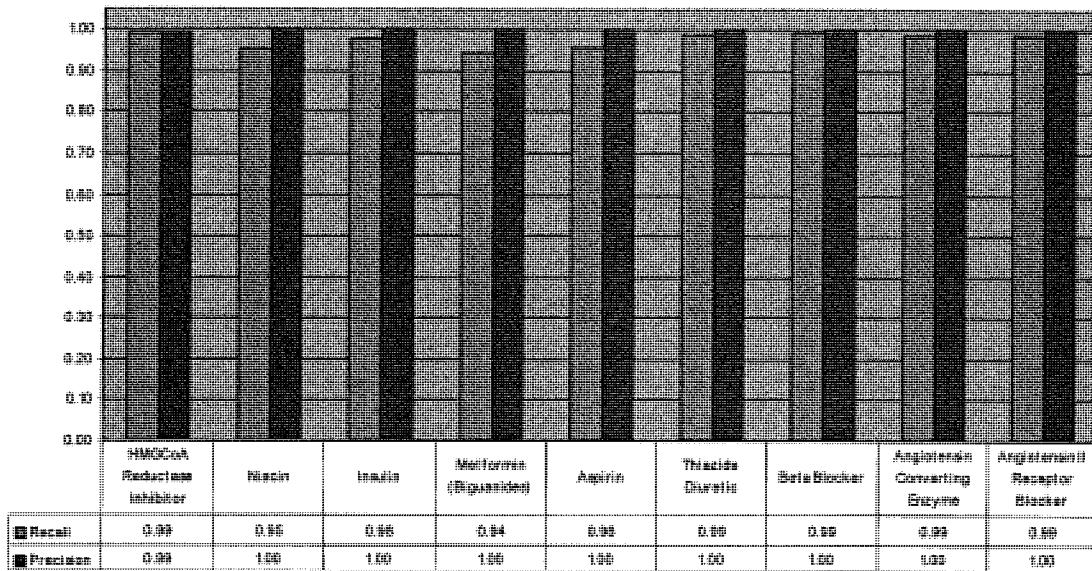


Figure 6. Recall and Precision for MEDICATION Concepts

## Phase II

This phase was designed to compare query processes that differed in their use of a hierarchical terminology system. The first task that study subjects were asked to complete was a term lookup exercise. The purpose of this task was to determine if the terminology browser would significantly influence the time it took to complete subsequent queries, as well as to gain experience with the two different term lookup methods. Each participant was asked to look up the concept code for *inflammatory bowel disease* (term 1) and *methotrexate* (term 2) using both the SNOMED CLUE Browser and the GE Terminology Tool. A sample screen shot can be seen in Figure 7. The mean lookup times are shown in Table 8. The mean difference was 3.7 seconds for term 1 (95% confidence interval (c.i.) -7.4 to 14.8) and 14.0 seconds for term 2 (95% c.i. -9.0 to 37.0). Though the GE lookup time was longer in both cases, the difference was not significant for an alpha of 0.05. There was also no significant difference in either the clinician or non-clinician subgroup as well as no significant difference in any lookup time between the two subgroups. Of the total of 48 terms looked up, only two were incorrect, one for each system.

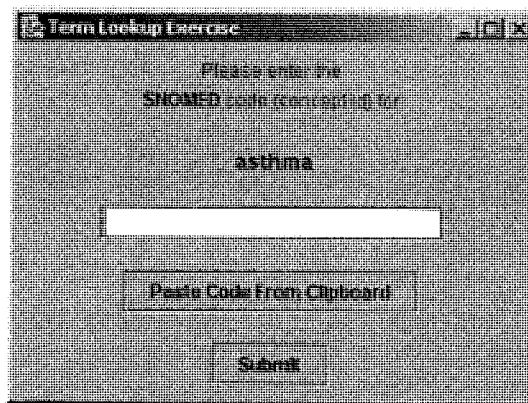


Figure 7. Term Lookup Example

	Mean Lookup Time (sec)
SNOMED Term 1	58
GE Term 1	62
SNOMED Term 2	43
GE Term 2	57

**Table 8. Term Lookup Results**

The next part was the main task used to evaluate the Phase II hypothesis. Each subject was asked to perform the following two queries:

1. Select patients with hyperlipidemia and type II diabetes mellitus who are not taking niacin (also known as nicotinic acid).
2. Select patients with coronary artery disease and hypertension who do not have heart failure.

Each query was done using each method. A summary of the results is shown in Table 9.

Query	Recall			Precision			Time (Seconds)		
	Mean	Lower 95 % C.I.	Upper 95 % C.I.	Mean	Lower 95 % C.I.	Upper 95 % C.I.	Mean	Lower 95 % C.I.	Upper 95 % C.I.
SNOMED 1	0.99	0.99	0.99	1.00	0.99	1.00	368	206	530
GE 1	0.40	0.30	0.50	0.89	0.71	1.07	761	629	893
SNOMED 2	0.65	0.43	0.87	0.93	0.85	1.01	302	242	362
GE 2	0.37	0.28	0.46	0.90	0.81	0.99	595	387	804

**Table 9. Phase II Recall, Precision, and Time**

Precision scores for each query were high, with means ranging from 0.89 to 1.00. Lower precision scores were most often caused by the subject either failing to add a concept for which patients were being excluded (e.g. niacin in Query 1), or failing to retrieve appropriate patients for those same concepts. For example, one subject, while looking for patients with heart failure, set up that concept query as a *medication* query instead of the correct *problem* query, and therefore did not retrieve any patients, and so did not remove patients with heart failure from the final retrieval set. These types of errors are

independent of the vocabulary system. As can be seen in Table 10, the differences in precision between the SNOMED and non-hierarchical queries as measured by a paired t-test were not significant.

	Mean of Difference	95% Confidence Interval of the Difference		p-value
		Lower	Upper	
Query 1 Recall	0.59	0.49	0.69	<0.005
Query 1 Precision	0.11	-0.07	0.28	0.22
Query 1 Time	-393	-533	-253	<0.005
Query 2 Recall	0.28	0.06	0.49	0.02
Query 2 Precision	0.04	-0.05	0.12	0.36
Query 2 Time	-293	-458	-128	<0.005

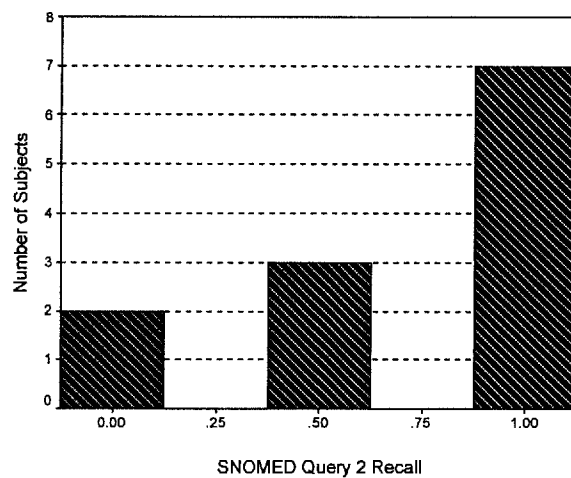
**Table 10. Phase II Paired t-test Results. The difference in each case is calculated: SNOMED – GE**

There was more variability with recall. Recall for the SNOMED queries behaved in a binary or indexed manner. If the subject selected the correct SNOMED concept id for the two inclusive concepts in each query, the recall was 0.99 for Query 1 and 0.90 for Query 2. If an incorrect concept id was entered, the recall was 0.00. In two cases, more specific concepts than the preferred concept were entered, which led to midrange recalls. One example of this was the use of the concept *essential hypertension* instead of the more general *hypertensive disorder*. Figure 8 demonstrates this phenomenon with a histogram, in which one can see that seven of twelve subjects chose the correct concepts for Query 2. With a mean recall of 0.99 for SNOMED Query 1, one can see that each subject found the appropriate concepts.

For the queries using the non-hierarchical system, the recall was lower, with a mean of 0.40 for Query 1 and 0.37 for Query 2. The highest individual recall using this system



for each query was 0.48 for Query 1 and 0.58 for Query 2. As seen in Table 10, the difference in recall between the two systems was significant at  $p < 0.005$  for Query 1 and  $p = 0.02$  for Query 2. This finding held true for each subgroup as well, with the exception of Query 2 recall in the clinician subgroup, where the mean difference was 0.20 with  $p = 0.32$ . There were also no significant differences in precision and recall with either query between the two subgroups when they were directly compared.<sup>4</sup>



**Figure 8. Histogram for SNOMED Query 2 Recall**

The low recall found with the non-hierarchical method raised the possibility that that method was not a realistic alternative to the SNOMED method. In order to address this issue, another GE analyst who had worked with MicroStrategy (MicroStrategy Inc., McLean, VA), an on-line analytical processing (OLAP) application, was asked to perform the same queries using that tool. Because aggregations for the diagnoses of *congestive heart failure*, *ischemic heart disease*, and *hypertension* had already been

---

<sup>4</sup> This is for an alpha of 0.05, and was measured with a t-test for independent samples.

created, performing Query 2 required only selecting those pre-aggregated concepts. The recall for that query was 0.63 with a precision of 0.98. For Query 1, there was an aggregation in place for *hyperlipidemia*, but not for the other two concepts. The analyst used a GPI stem for the concept of *niacin*, much like the strategy used in medication queries in Phase I. Although there was an aggregation in place for *diabetes mellitus* in general, there was none specifically for *type II diabetes mellitus*. The analyst had the option of aggregating ICD-9 codes, problem strings, or concept descriptions to describe the concept. She decided to collect concept descriptions, much like the process used with the non-hierarchical query system. The recall for this query was 0.09 with a precision of 0.99. This low value occurred due to low retrievals for both *type II diabetes mellitus* and *hyperlipidemia*.

The time taken to complete each query also differed significantly between the two methods, with Query 1 taking 393 seconds longer using the non-hierarchical system, and Query 2 taking 293 second longer using that method. This finding also held true for each subgroup.

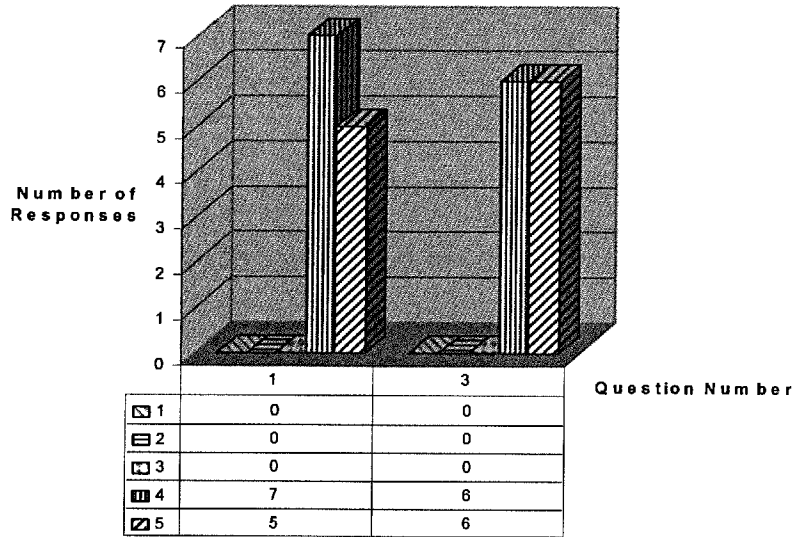
Query Concept	Maximum	Mean
Hyperlipidemia	16	11
Type II Diabetes Mellitus	37	30
Niacin	66	39
Coronary Artery Disease	10	4
Hypertension	49	32
Heart Failure	13	9

**Table 11. Number of GEMS concepts used by test subjects to express query concept**

Each query concept was expressed by a single SNOMED concept, but required multiple GEMS concepts for expression. The number of GEMS concepts required to express each

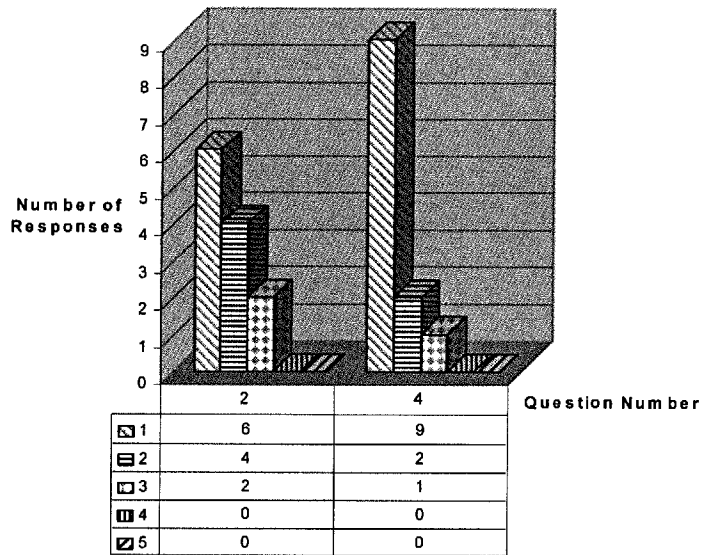
query concept is shown in Table 11. Subjects selected up to 66 GEMS concepts to express a query concept. The overall mean number of GEMS concepts needed to express each query concept was 21. One should also realize that even this relatively large number of GEMS concepts still produced rather poor recall. For comparison, when SNOMED queries were constructed correctly, a mean of 31 GEMS concepts were selected for each SNOMED concept.

In the post-trial questionnaire, the participants found the method utilizing SNOMED easier to use and reported they were more likely to use the SNOMED method in the future. These results can be seen in Figure 9 and Figure 10. Higher-numbered responses are more positive. For the two questions regarding SNOMED, questions 1 and 3, all the responses were 4s and 5s, with mean scores of 4.33 and 4.50, respectively. For the two questions regarding the non-hierarchical method, questions 2 and 4, all the responses were 1s, 2s and 3s, with mean scores of 1.67 and 1.33, respectively. The differences between the means for questions 1 and 2, as well as questions 3 and 4, were significant, with  $p < 0.005$ .



**Figure 9. Questionnaire Responses to SNOMED Questions**

1. Please rate ease of use for the SNOMED method (1 – very difficult, 5 – very easy).
2. How likely would you be to use the SNOMED method (1 – very unlikely, 5 – very likely)?



**Figure 10. Questionnaire Responses to Non-hierarchical Questions**

2. Please rate ease of use for the non-hierarchical method (1 – very difficult, 5 – very easy).
4. How likely would you be to use the non-hierarchical method (1 – very unlikely, 5 – very likely)?

# Discussion

## *Overview*

The evidence presented strongly supports the hypotheses for both Phase I and Phase II of this study. In Phase I, it was demonstrated that a query based on SNOMED was easier to create and maintain than an ad-hoc query created without the use of SNOMED. It was also shown that there was no significant degradation of results as measured by recall and precision in performing the SNOMED-based queries. In Phase II, queries created using a SNOMED-based tool had a significantly higher recall and shorter query time compared to queries using a non-hierarchical vocabulary based tool.

This study demonstrated that a query system utilizing a hierarchically-organized, concept-based reference terminology allowed end users to achieve high levels of recall and precision when attempting difficult ad-hoc queries by leveraging the terminology's incorporated semantic knowledge. The results illustrate the benefits that could be derived by following the course of integrating an external reference terminology into the query process. The major cost associated with such a course was also demonstrated by the length of time required to do the initial mapping. One would likely encounter these same costs and benefits, albeit on a much larger scale, if this type of system were implemented in a production system. Though the initial mapping would have high costs, the ongoing benefits of easier query creation and maintenance, and higher quality query results, would likely outweigh those costs.

## ***Relationship to Previous Work***

Like the system developed by Partners Healthcare, the SNOMED-based query tool also made use of hierarchically-organized problem and medication concepts[27]. Unlike the Partners Healthcare hierarchies that were developed and maintained in-house, this study took advantage of the hierarchical relationships that are part of SNOMED. Because the Partners Healthcare system is a production system, and the study system only dealt with a limited domain, it is difficult to compare the level of effort required to maintain each system. However, if a production system were developed around SNOMED, maintenance would involve only two tasks: mapping new concepts to SNOMED and updating the mapping table with SNOMED updates. As mapping of text strings to new GEMS concepts is already being done, the additional task of mapping that GEMS concept to a SNOMED concept should not take much time. Overall, maintenance of the SNOMED-based system should take considerably less resources than the full-time analyst required for the maintenance of the Partners Healthcare System[13].

The results from both phases of this study confirmed the results from Stoffel, et al. regarding the simplification of queries through the use of a hierarchical vocabulary system[9]. That study described how a query involving over 300 attributes could be expressed with only two attributes using their hierarchical vocabulary based tool. Phase I demonstrated that the reference MQIC query that required multiple inclusion and exclusion criteria could be expressed with a single SNOMED concept. Phase II revealed that a single SNOMED concept could be substituted for an average of 31 GEMS concepts.

Unlike previous work in this area, with the exception of the study noted above, Phase II of this trial used quantitative methods to measure the impact of using a hierarchically-organized concept-based reference terminology on the querying of a clinical data warehouse. In doing so, the results from Phase II demonstrated that the use of such a reference terminology significantly improves the ability of non-experts to retrieve useful and meaningful information from a data warehouse.

## ***Limitations***

### **Phase I**

The design and execution of this study did have several limitations. One of the purposes of Phase I was to evaluate the performance of SNOMED. There were other factors other than SNOMED's completeness and correctness that could have affected the outcome of Phase I. By using only the *Is-A* relationship, all of the information contained in SNOMED was not being accessed and all relevant concepts may not have been retrieved. For example, in looking for patients with diabetes, one would have wanted to find any patient with *diabetic neuropathy*. However, this concept is not a sub-type of diabetes in the *Is-A* hierarchy in SNOMED, but rather is connected through the *Has-Associated-Etiology* relationship. Nevertheless, limiting query construction to the *Is-A* relationship was reasonable in that part of the hypothesis from Phase I was that a hierarchical terminology makes creating queries simpler, which was less likely to be true if one had to use multiple hierarchies.

The quality and completeness of mapping also had a great effect on the outcome of this phase. Failure to map GEMS terms to appropriate SNOMED concepts that are returned by a query would have led to failure to retrieve the appropriate patients and would weaken the SNOMED result. Using only pre-coordinated concepts could also have exacerbated this problem. As mentioned in the previous sections, only two mapping errors were uncovered that caused a small decrease in the precision for the concept *HMGCoA reductase inhibitor*, and a small decrease in the recall for the concept *CAD*.

The choice of a limited clinical domain could also have affected outcomes. For example, SNOMED might have fewer post-coordinated concepts in the area of cardiovascular disease, which would have impacted the mapping and eventually the concordance between the two methods. Likewise, it is possible that the area selected for study provided excellent results, but might not be representative of all of SNOMED. With limited resources, this limitation was unavoidable. The solution to this problem would be to map the entire GEMS terminology and perform a wide array of queries.

Finally, the precision and recall could have been lowered, not because the SNOMED query was failing, but rather, because the SNOMED query was actually performing in a superior manner in identifying appropriate patients. Examining preliminary results and improving the MQIC query when necessary addressed this possibility. This revision of MQIC queries was reasonable because the MQIC query was used as a reference to evaluate the SNOMED query and therefore needed to be as accurate as possible. In addition, the concepts selected in the MQIC query that were not selected in the



SNOMED query were examined and found to be appropriate for the given query, thereby verifying the recall results.

## **Phase II**

There were also limitations in Phase II of the study. While using different browsers for the two methods was necessary, it could also have confounded the experiment. Because the intent of this phase was to test the usefulness of the hierarchical nature of SNOMED, the browser had to have hierarchical properties. However, the usability of the browser, other than its hierarchical nature, could have affected the query results. In order to address this possibility, the term lookup exercise was added to measure how long it took the subjects to find specific codes for concepts using each browser, and whether they retrieved the correct code. Although the times were not significantly different, they were shorter for the SNOMED system.

A second factor that could have affected the results was the clinical expertise of the subjects. A clinician may have been better able to find all relevant concepts in the GEMS terminology or the one most appropriate concept in SNOMED. This difference in clinical ability could have influenced one method more than the other. Subjects with a mix of clinical experiences were recruited to address this possible problem and did not find a significant difference in the performance of these two groups.

Finally, the non-hierarchical system used for comparison might not have been a realistic comparison method. Non-expert analysts might be able to routinely concoct methods of ad-hoc querying that return better results than the tested system. The observation of the

GE analyst using MicroStrategy was done to address this possibility. The mean recalls for the two queries using the non-hierarchical system (0.40 and 0.37) compared favorably to the recalls obtained using MicroStrategy (0.09 and 0.63), which supported the use of the non-hierarchical system as a reasonable comparison for the SNOMED system. The MicroStrategy results also demonstrated the difficulty of performing ad-hoc queries. Furthermore, observation of the analyst's method of aggregating terms using GEMS concepts lent credibility to the use of this same method as part of the non-hierarchical query system.

## **Conclusion**

The process of improving healthcare quality and safety requires accurate retrieval of clinical data for various purposes such as population-based management of high-cost preventable diseases, decision support using evidence-based guidelines, outcomes research, measurement of healthcare delivery system performance, and the conducting and evaluation of clinical trials. The results of this study have demonstrated how the use of a hierarchically-organized, concept-based reference terminology has improved the efficiency and accuracy of information retrieval.

## References

1. Ricciardi, T.N., F.E. Masarie, and B. Middleton, *Clinical benchmarking enabled by the digital health record*. Medinfo, 2001. **10**(Pt 1): p. 675-9.
2. van Mulligen, E.M., H. Stam, and A.M. van Ginneken, *Clinical data entry*. Proc AMIA Symp, 1998: p. 81-5.
3. Cimino, J.J., et al., *Managing vocabulary for a centralized clinical system*. Medinfo, 1995. **8 Pt 1**: p. 117-20.
4. Rose, J.S., et al., *Common medical terminology comes of age, Part One: Standard language improves healthcare quality*. J Healthc Inf Manag, 2001. **15**(3): p. 307-18.
5. Cimino, J.J., *Desiderata for controlled medical vocabularies in the twenty-first century*. Methods Inf Med, 1998. **37**(4-5): p. 394-403.
6. Chute, C.G., S.P. Cohn, and J.R. Campbell, *A framework for comprehensive health terminology systems in the United States: development guidelines, criteria for selection, and public policy implications*. ANSI Healthcare Informatics Standards Board Vocabulary Working Group and the Computer-Based Patient Records Institute Working Group on Codes and Structures. J Am Med Inform Assoc, 1998. **5**(6): p. 503-10.
7. Forman, B.H., et al., *Applying a controlled medical terminology to a distributed, production clinical information system*. Proc Annu Symp Comput Appl Med Care, 1995: p. 421-5.
8. Huff, S.M., et al., *Panel: The Impact of Business Issues on Terminology Adoption: Clinical Software Developer's Perspective*. Proc AMIA Symp, 1999.
9. Stoffel, K., et al., *A graphical tool for ad hoc query generation*. Proc AMIA Symp, 1998: p. 503-7.
10. Cimino, J.J., *From data to knowledge through concept-oriented terminologies: experience with the Medical Entities Dictionary*. J Am Med Inform Assoc, 2000. **7**(3): p. 288-97.
11. Gu, H., et al., *Benefits of an object-oriented database representation for controlled medical terminologies*. J Am Med Inform Assoc, 1999. **6**(4): p. 283-303.
12. Murphy, S.N., et al., *Optimizing healthcare research data warehouse design through past COSTAR query analysis*. Proc AMIA Symp, 1999: p. 892-6.
13. Murphy, S.N., *Personal Communication*.
14. Deshpande, A.M., C. Brandt, and P.M. Nadkarni, *Metadata-driven ad hoc query of patient data: meeting the needs of clinical studies*. J Am Med Inform Assoc, 2002. **9**(4): p. 369-82.
15. Nadkarni, P.M., C.M. Brandt, and L. Marengo, *WebEAV: automatic metadata-driven generation of web interfaces to entity-attribute-value databases*. J Am Med Inform Assoc, 2000. **7**(4): p. 343-56.
16. Nadkarni, P.M. and C. Brandt, *Data extraction and ad hoc query of an entity-attribute-value database*. J Am Med Inform Assoc, 1998. **5**(6): p. 511-27.
17. Lau, L.M., et al., *Enhancing an Enterprise Data Warehouse with a Data Dictionary*. Proc AMIA Symp, 2001: p. 951.

18. Lau, L.M., et al., *A method for the automated mapping of laboratory results to LOINC*. Proc AMIA Symp, 2000: p. 472-6.
19. *Medscape Quality Improvement Consortium Prospectus*. 2001, Medscape: Hillsboro, OR.
20. *SNOMED Clinical Terms*. 2002, College of American Pathologists: Northfield, IL.
21. Rose, J.S., et al., *Common medical terminology comes of age, Part Two: Current code and terminology sets--strengths and weaknesses*. J Healthc Inf Manag, 2001. **15**(3): p. 319-30.
22. *SNOMED Clinical Terms, Technical Implementation Guide*. 2002, College of American Pathologists: Northfield, IL.
23. Campbell, J.R., et al., *Phase II evaluation of clinical coding schemes: completeness, taxonomy, mapping, definitions, and clarity*. CPRI Work Group on Codes and Structures. J Am Med Inform Assoc, 1997. **4**(3): p. 238-51.
24. Chute, C.G., et al., *The content coverage of clinical classifications. For The Computer-Based Patient Record Institute's Work Group on Codes & Structures*. J Am Med Inform Assoc, 1996. **3**(3): p. 224-33.
25. Elkin, P.L., et al., *A randomized controlled trial of the accuracy of clinical record retrieval using SNOMED-RT as compared with ICD9-CM*. Proc AMIA Symp, 2001: p. 159-63.
26. Lussier, Y.A. and M. Bourque, *Comparing SNOMED and ICPC retrieval accuracies using relational database models*. Proc AMIA Annu Fall Symp, 1997: p. 514-8.
27. Murphy, S.N., G.O. Barnett, and H.C. Chueh, *Visual query tool for finding patient cohorts from a clinical data warehouse of the partners HealthCare system*. Proc AMIA Symp, 2000: p. 1174.

## Appendix 1: Query Tool Instructions

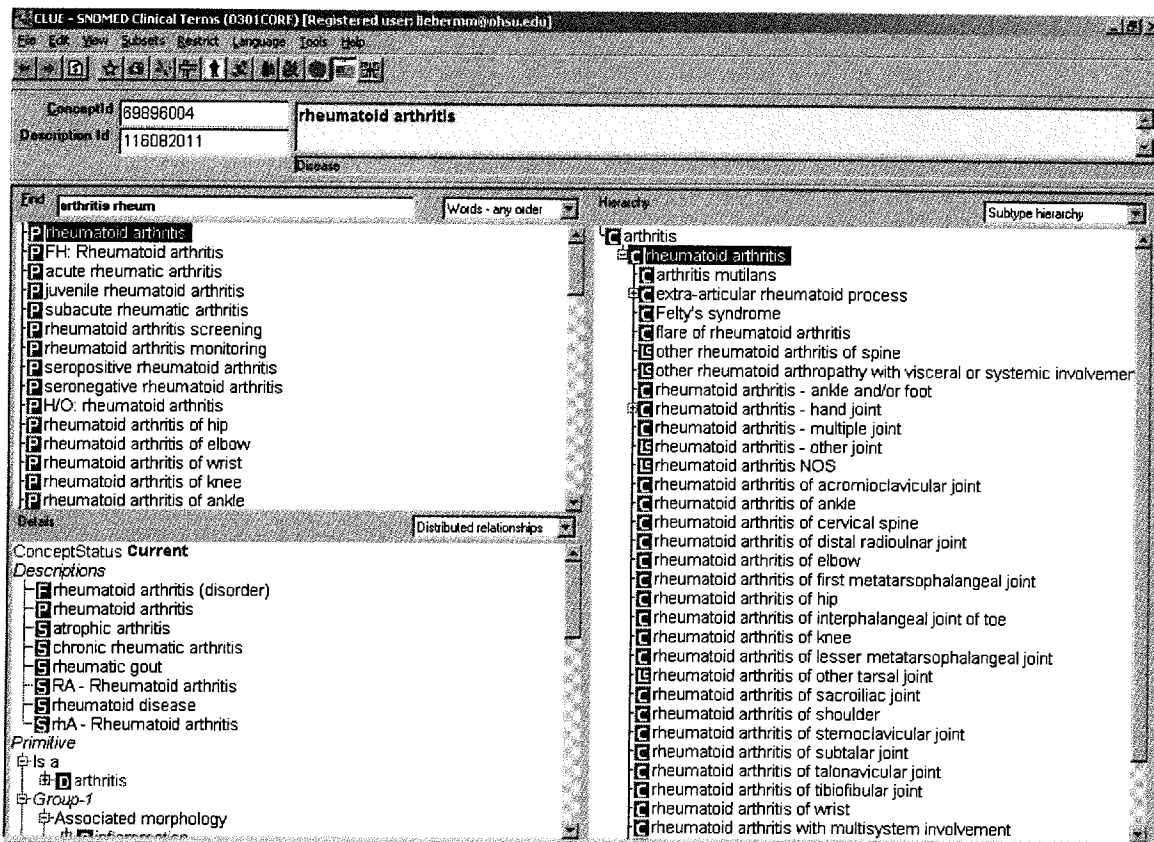
## Introduction

Thank you for agreeing to participate in this study. The overall goal of the study is to compare how two different terminology systems affect querying of a clinical data warehouse. You will perform the same queries (using the same query tool) using both systems. You will be using different look-up tools for the two different systems. In order to determine how much of an effect the look-up tools might have on the overall results, your first task will be to retrieve individual concept codes using the two different lookup tools. Next, you will be asked to build complex queries to answer clinically relevant questions.

## Part 1, Term Lookup

For both methods, to input the selected code into the form, you can either type it in yourself, use **ctrl-v** to paste it, or click the **Paste Code From Clipboard** button.

## SNOMED CLUE Browser



Appendix 1, Figure 1

To find the SNOMED concept code, type all or part of a phrase in the text box to the right of the *Find* label. A list of concept descriptions matching your search terms will appear below that box. Scroll through the list and click on a concept that is at least close to what you are looking for. The preferred term for that concept will appear in its place in SNOMED's hierarchy in the window to the right. You can then select a more specific or less specific term in the hierarchy by a single click on that term in the right window. A double click of a term in the hierarchy window will reset that window to include both more specific and less specific terms. In the box at the bottom left of the window, you can see synonyms of the selected term under the *Descriptions* heading. If you are unsure whether a concept is exactly the one you are looking for, it may help to click on it, and then check the synonyms.

If you are having trouble finding a term, you might try selecting a more specific concept (i.e. ibuprofen 800mg tablet) in the results of the search, then move up the hierarchy in the right hand box to choose the more general concept (ibuprofen). When you have selected the term you wish to retrieve, double click on the text field to the right of the **ConceptId** label to highlight the code (you can also click and drag over this number to highlight it). Then copy the code to your clipboard using either **ctrl-c**, or a right click followed by the copy selection.

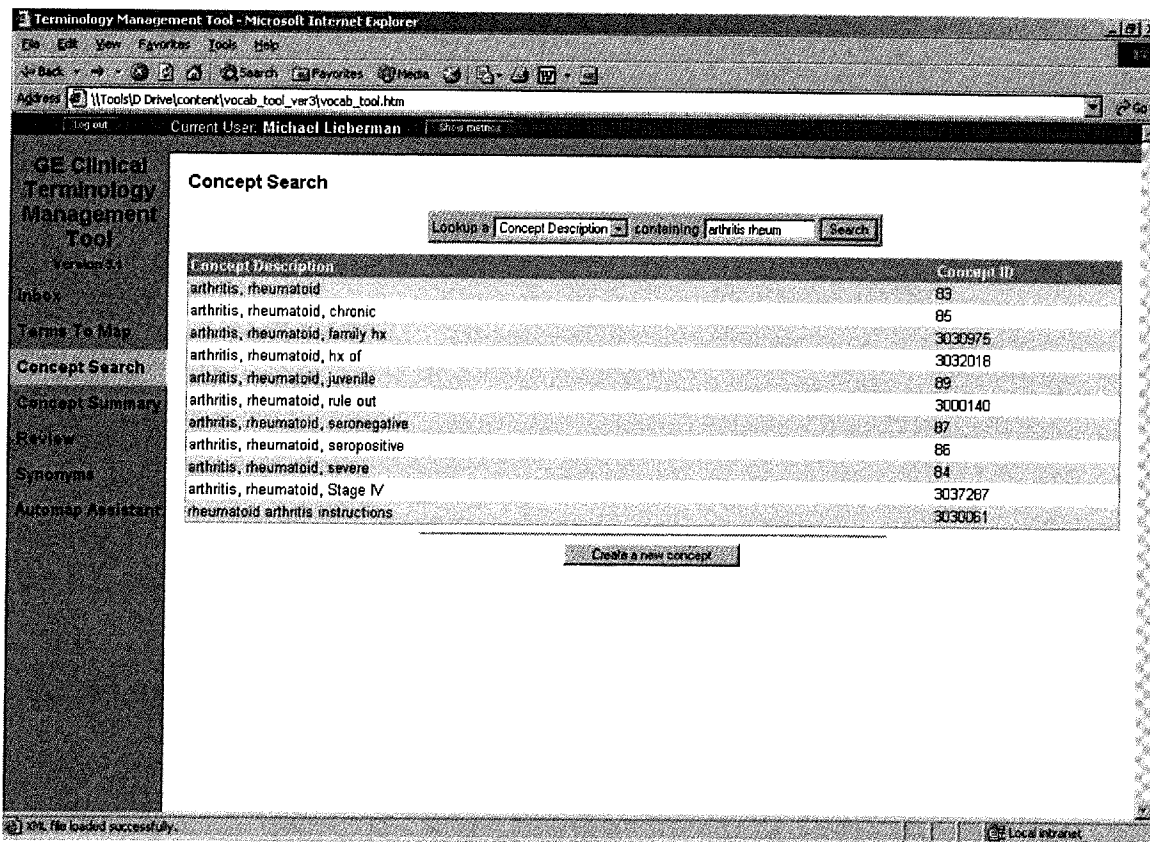
Selecting medication concepts can be confusing. SNOMED lists medications in two different hierarchies, substances and products, which have different concept codes. For the purpose of this study, you will always want to choose the product. The concept name is displayed in the large box with a yellow background to the right of the **ConceptId** field. The box directly below this box displays which hierarchy the concept belongs to.

When you are looking up a medication, be sure this box contains the phrase:

**' Pharmaceutical/biologic product'** and not **'Substance'**.

When you are looking up problems, this box should contain the word: **Disease**.

# GE Terminology Concept Search



Appendix 1, Figure 2

This tool uses Internet Explorer. You should see the label **Concept Search** at the top left of the window. **Concept Description** should be displayed in the drop down box to the right of the **Lookup a** label. Type all or part of the word or phrase in the field to the right of the **containing** label and click the **Search** button. The Concept Descriptions and Concept Ids (codes) will then be displayed below. Find the Concept Description you are looking for, and simply click (left) on the corresponding Concept ID to copy it to the clipboard. If you click on the Concept Description by mistake you will be taken to a new page. To return to the search results, you need to move the browser window to expose the left hand panel, and click on **Concept Search**.

## Part 2, Queries

You will be asked to perform each query twice; once using SNOMED to select codes to express a concept, and once using the GE vocabulary to do the same. The major difference between the two systems is that SNOMED contains relationships between concepts implemented through multiple hierarchies while the GE vocabulary does not contain these relationships. The query tool that you will use takes advantage of these



SNOMED relationships. When you select a SNOMED concept term, that term will be expanded to include all more specific concepts. For example, if you select arthritis, all more specific types of arthritis, such as rheumatoid arthritis and osteoarthritis, will also be selected. This type of expansion will not occur when using the GE system.

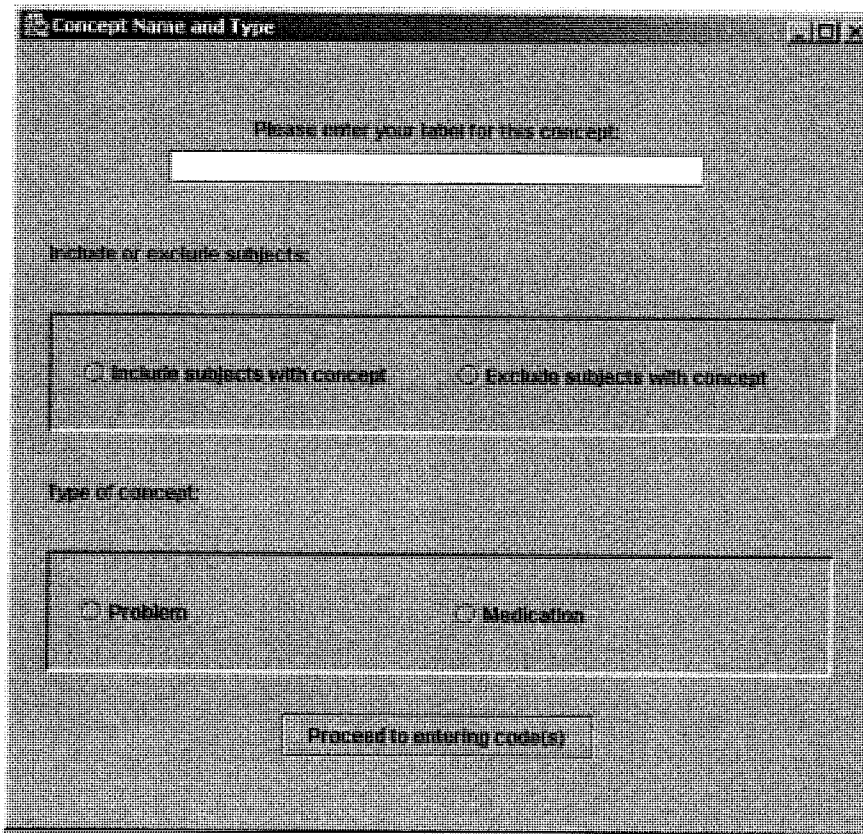
## Main Query Screen

The screenshot shows a window titled "Query #2g" with a dark background. At the top center, it says "Query Method: GEMS". Below that, the query question is displayed: "Select all patients congestive heart failure not on an ACE inhibitor." There are two buttons: "Add New Concept" on the left and "Query Complete" on the right. Below these are two columns for including and excluding subjects. Each column has a header "Include Subjects With:" or "Exclude Subjects With:" and two sub-headers: "Concept" and "Patients with Concept". There are empty text boxes under each of these sub-headers. At the bottom, it says "Number of currently selected patients: 0" with a small box next to the number.

Appendix 1, Figure 3

The Main Query Screen will display the query question, such as “**Select all patients with myocardial infarction who are not taking ACE inhibitors,**” as well as which system you are to use to answer the query: **GEMS** or **SNOMED**. The queries are concept based. As concepts are entered, the results for each concept will be displayed as described below. After all concepts for a particular query have been entered, you will click the **Query Complete** button to finish the query. For the example question stated above, you will first need to find all patients with myocardial infarction, and then remove all of those that are taking ACE inhibitors. To begin the query, press the **Add New Concept** button.

## Concept Description Screen

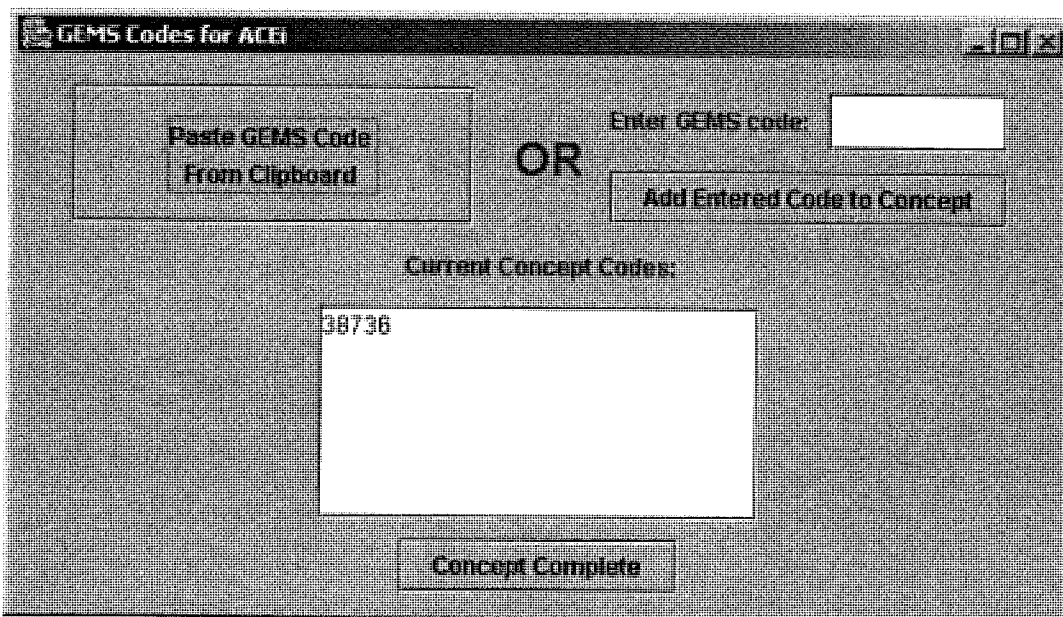


The screenshot shows a window titled "Concept Name and Type". It contains a text input field with the prompt "Please enter your label for this concept:". Below this is a section titled "Include or exclude subjects:" with two radio buttons: "Include subjects with concept" and "Exclude subjects with concept". The next section is titled "Type of concept:" with two radio buttons: "Problem" and "Medication". At the bottom is a button labeled "Proceed to entering code(s)".

Appendix 1, Figure 4

The Concept Description Screen will be displayed. You first need to enter a label for the concept. This label can be anything of your choosing (though try to keep it short due to the limited space in which it will be displayed). Its purpose is to label the concept so that you can keep track of it. For “myocardial infarction,” you might use “MI.” The next box asks you to specify whether you want to include or exclude patients with the concept. In the previous example, because the query asks for “all patients *with* myocardial infarction”, you would select **include**. For the second part of that query, “... who are *not* taking ACE inhibitors,” when starting the concept “ACE inhibitor” you should select **exclude** because of the “not.” In the next field, you need to specify whether the concept is a problem or a medication, so that the correct database table can be queried. Click on either the **Problem** or **Medication** button. When you are finished, click on **Proceed to entering code(s)**.

## Concept Code Entering Screen



Appendix 1, Figure 5

The next screen is the Concept Code Entering Screen. This screen is used to enter either the SNOMED code or GE codes for the concept. Be sure you use the appropriate concept lookup tool as specified on the Main Query Screen. Once you have looked up the concept using the appropriate tool, and copied the concept ID to the clipboard, click on the **Paste Code From Clipboard** button to enter the code. Continue to lookup and enter codes. When you are finished looking up codes, click on the **Concept Complete** button. You will notice that the button initially turns and remains dark gray. It will remain this color until the results for this concept come back from the database; a process that can take up to 30-40 seconds. When the results do come back, the Concept Code Entering Screen will disappear. The results will be displayed on the Main Query Screen with the label you assigned, the number of patients selected, and a button that can be used to remove that concept from your query. If the concept returns 0 patients, it is likely that on the Concept Description Screen, you chose medication when you should have chosen problem, or vice versa. To correct the error, you will need to remove the concept from the query, and start the concept over again. You then have the option of either adding another concept to the query with the **Add New Concept** button, or finishing the query with the **Query Complete** button. The use of the Concept Code Entering Screen differs depending on which vocabulary system you are using.

- SNOMED – In general, because of SNOMED’s hierarchical structure, you should only need to select one term to express a concept.
- GE – Because the GE system does not include relationships between concepts, you will need to enter **all** the codes for terms that express a concept.

## Appendix 2: SNOMED Concept – GPI Stem Mapping

Med/Class	GPI Stem	SNOMED	SNOMED Term
insulins	271	39487003	Insulin product
alpha-glucoside inhibitors	275	109072000	Alpha-glucoside inhibitor
biguanides	2725	109082004	Biguanide
sulfonylureas	2720	34012005	Sulfonylurea
meglitinides	2728	109075003	Meglitinide
thiazolidinediones	27607	346597008	Oral hypoglycemic
HMG CoA reductase inhibitors	3940	96302009	HMG-CoA reductase inhibitor
anion exchange resins	391	346322006	Anion exchange resin
bile acid sequestrants	391	83750004	Bile acid sequestrant antilipemic agent
probucol	395	32474005	Probucol
gemfibrozil	3920003	35282000	Gemfibrozil
clofibrate	3920001	77035009	Clofibrate
fenofibrate	3920002	108603001	Fenofibrate
nicotinic acids	3945	63639004	Niacin preparation
aspirin	6410001	7947003	Aspirin
warfarin	832	48603004	Warfarin
beta blocking agents, non-	331	83522001	Non-selective beta-blocking agent
beta blocking agents, beta 1	332	15772006	Beta 1 blocking agent
alpha <b>beta</b> blocking agents	333	83522001	Non-selective beta-blocking agent
<b>alpha</b> beta blocking agents	333	76385003	Alpha 1 adrenergic blocking agent
calcium channel blockers	34	48698004	Calcium channel blocking agent
ACE inhibitors	3610	41549009	Angiotensin-converting enzyme inhibitor
angiotensin II receptor antagonists	3615	96308008	Angiotensin II receptor antagonist
alpha-2 agonists	36201	108828006	Alpha 2 adrenergic agonist
doxazosin	36202005	108556006	Doxazosin
guanadrel	3620201	372820001	Guanadrel
guanethidine	3620202	79305004	Guanethidine
prazosin	3620203	76058001	Prazosin
terazosin	3620204	129484001	Terazosin
reserpine	36203	78379001	Reserpine
nonspecific alpha blocking agent	363	373251009	Nonspecific alpha-adrenergic blocking agent
vasodilators	36400	318641000	Vasodilator antihypertensive drugs
fenoldopam	364020	108590002	Fenoldopam
<b>reserpine</b> / thiazides	36991002	78379001	Reserpine
reserpine / <b>thiazides</b>	36991002	45518007	Thiazide diuretic
<b>reserpine</b> / thiazides / hydralazine	36991003	78379001	Reserpine
reserpine / <b>thiazides</b> / hydralazine	36991003	45518007	Thiazide diuretic

Med/Class	GPI Stem	SNOMED	SNOMED Term
<b>ACE inhibitors / calcium channel</b>	369915	41549009	Angiotensin-converting enzyme inhibitor
ACE inhibitors / <b>calcium channel</b>	369915	48698004	Calcium channel blocking agent
<b>ACE inhibitors / thiazides</b>	369918	41549009	Angiotensin-converting enzyme inhibitor
ACE inhibitors / <b>thiazides</b>	369918	45518007	Thiazide diuretic
<b>beta blocking agents / thiazides</b>	36992	33252009	beta-Blocking agent
beta blocking agents / <b>thiazides</b>	36992	45518007	Thiazide diuretic
<b>angiotensin II receptor</b>	36994	96308008	Angiotensin II receptor antagonist
angiotensin II receptor antagonists	36994	45518007	Thiazide diuretic
<b>alpha-2 agonists / thiazides</b>	369950	108828006	Alpha 2 adrenergic agonist
alpha-2 agonists / <b>thiazides</b>	369950	45518007	Thiazide diuretic
<b>prazosin / thiazides</b>	36995502	76058001	Prazosin
prazosin / <b>thiazides</b>	36995502	45518007	Thiazide diuretic
<b>vasodilators / thiazides</b>	36999	318641000	Vasodilator antihypertensive drugs
vasodilators / <b>thiazides</b>	36999	45518007	Thiazide diuretic
potassium sparing diuretics	375	13929005	Potassium sparing diuretics
thiazide diuretics	376	45518007	Thiazide diuretic
<b>potassium sparing diuretics /</b>	3799	13929005	Potassium sparing diuretics
potassium sparing diuretics /	3799	45518007	Thiazide diuretic

# Appendix 3: Phase II Consent Form

IRB# 7385-1  
Protocol Approved: 12/6/02

**OREGON HEALTH & SCIENCE UNIVERSITY**  
**Informed Consent Form**

**TITLE:** The Use of SNOMED to Enhance Querying of a Clinical Information Data Warehouse. Stage II.

**PRINCIPAL INVESTIGATOR:** Michael Lieberman, MD (503) 494-3731

**CO-INVESTIGATOR(S):** None

**SPONSOR:** Kent Spackman, MD PhD

**PURPOSE:**

You have been invited to participate in this research study because of your knowledge of databases and SQL. The purpose of this study is to evaluate whether or not the use of SNOMED can improve querying of a clinical data warehouse.

Your participation in this study will last two hours.

6-10 subjects will be enrolled in the study.

**PROCEDURES:**

You will be given instructions in two different methods of querying the database. After instruction in the first method, you will be asked to perform a set of queries, and the result sets will be recorded. You will then receive instruction on the second method, and be given a second set of queries to perform.

Each participant will use both methods to query the database. The order of the methods will be randomly assigned, with half the subjects using one method first, and the other half using the other method first.

The principle investigator will know the order of the methods used. **RISKS AND DISCOMFORTS:** None

Rev. 03/01/02



**BENEFITS:**

You may or may not personally benefit from participating in this study. However, by serving as a subject, you may contribute new information which may benefit patients in the future.

**ALTERNATIVES:**

You may choose not to participate in this study.

**CONFIDENTIALITY:**

Neither your name nor your identity will be used for publication or publicity purposes.

**COSTS:**

There are no costs to participate in this study.

Participants will receive a \$30 honorarium to participate in the study.

**LIABILITY:**

The Oregon Health & Science University is subject to the Oregon Tort Claims Act (ORS 30.260 through 30.300). If you suffer any injury and damage from this research project through the fault of the University, its officers or employees, you have the right to bring legal action against the University to recover the damage done to you subject to the limitations and conditions of the Oregon Tort Claims Act. You have not waived your legal rights by signing this form. For clarification on this subject, or if you have further questions, please call the OHSU Research Integrity Office at (503) 494-7887.

**PARTICIPATION:**

Michael Lieberman ((503) 494-3731) has offered to answer any other questions you may have about this study. If you have any questions regarding your rights as a research subject, you may contact the OHSU Research Integrity Office at (503) 494-7887

You do not have to join this or any research study. If you do join, and later change your mind, you may quit at any time. If you refuse to join or withdraw early from the study, there will be no penalty or loss of any benefits to which you are otherwise entitled.

The participation of OHSU students in OHSU research is completely voluntary and you are free to choose not to serve as a research subject in this protocol for any reason. If you do elect to participate in this study, you may withdraw from the study at any time without affecting your relationship with OHSU, the investigator, the investigator's department, or your grade in any


Rev. 03/01/02

course.

**SIGNATURES:**

You will be provided with a copy of this form.

Your signature below indicates that you have read the foregoing and agree to participate in this study.

OREGON HEALTH & SCIENCE UNIVERSITY INSTITUTIONAL REVIEW BOARD PHONE NUMBER (503) 494-7887. CONSENT FORM APPROVAL DATE
<div style="border: 1px solid black; width: 100px; height: 40px; margin: 0 auto;"></div>
DEC 6 2002
APPROVED BY: 
Do Not Sign This Form After The Expiration Date Of: 12/5/03

Participant

Date

Investigator

Date

## Appendix 4: Query Tool Application Code

```

public void timeStmp(String queryNumber, String queryType) {
    pw.println("Subject#"+subject+"Query#"+
        queryNumber+" Query Type: "+queryType+
        " started at "+System.currentTimeMillis()+" milliseconds");
}

private void loadOrder(List subjectNumber) {
    orderList = new ArrayList();
    try {
        RandomAccessFile orderFile = new RandomAccessFile(orderF
            fileName, "r");
        do {
            st = new StringTokenizer(orderFile.readLine());
        } while (!Integer.parseInt(st.nextToken()) != subjectNumber);
        while (st.hasMoreTokens()) {
            nextQuerySpec = new String(2);
            nextQuerySpec[0] = st.nextToken();
            nextQuerySpec[1] = st.nextToken();
            orderList.add(nextQuerySpec);
        }
    } catch (IOException ioe) {System.out.println("IO error - loadOrder");
    }

private void loadText() {
    queryQuestions = new ArrayList();
    try {
        RandomAccessFile textFile = new RandomAccessFile(textFile
            eName, "r");
        String nextLine = textFile.readLine();
        do {
            queryText = new String(2);
            queryText[0] = nextLine;
            queryText[1] = textFile.readLine();
            nextLine = textFile.readLine();
            queryQuestions.add(queryText);
        } while (nextLine != null);
        textFile.close();
    } catch (IOException ioe) {System.out.println("IO error - loadText");
    }

private void printOrder(int subjectNumber) {
    Iterator it = orderList.iterator();
    while (it.hasNext()) {
        nextQuerySpec = (String[])it.next();
        System.out.println(nextQuerySpec[0] + " " + nextQuerySpe
            c[1]);
    }

private void printText() {
    Iterator it = queryQuestions.iterator();
    while (it.hasNext()) {
        queryText = (String[])it.next();
        System.out.println(queryText[0] + " " + queryText[1]);
    }

private void onCreateDisplay (String subject) {
    content.setLayout(new BorderLayout_Y_AXIS);
    content.add(Box.createVerticalGlue());
    JLabel welcomeLabel = new JLabel("Welcome Subject#"+subject);
}
    
```

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import java.io.*;
import javax.swing.*;

public class Session extends JFrame implements ActionListener {
    private String subject, orderFileName, textFileName, dbParamName;
    private Connection con;
    private String[] queryText, nextQuerySpec;
    private StringTokenizer st;
    private JButton startButton, queryQuestions, queryArray;
    private JButton startButton = new JButton("Start");
    private Query nextQuery;
    private Iterator queryIterator;
    private PrintWriter pw;
    private Container content;
    private JFrame temp = new JFrame("Database Query Study");

    public Session(String name) {
        super(name);
        Dimension screenSize = getToolkit().getScreenSize();
        orderFileName = "queryOrder.txt";
        textFileName = "queryText.txt";
        dbParamName = "dbParams.txt"; //file with driver on first line
        URL on 2nd, user on 3rd, password on 4th
        content=this.getContentPane();
        temp.setBounds(screenSize.width*48/100, 0, screenSize.width*52/100
            , screenSize.height);
        temp.setVisible(true);
        temp.setBackground(Color.black);
        temp.setBounds(662, 300, 300, 130);
    }

    public void actionPerformed(ActionEvent e) {
        queryIterator = queryArray.iterator();
        while (queryIterator.hasNext()) {
            this.display(queryIterator.next());
            CodeFind cf = new CodeFind(((String[])orderList.get(0))[1], codeL
                istener, pw);
        }

    public void initDB() {
        try {
            RandomAccessFile dbParams = new RandomAccessFile(dbParam
                sName, "r");
            driver = dbParams.readLine();
            url = dbParams.readLine();
            user = dbParams.readLine();
            password = dbParams.readLine();
            dbParams.close();
        } catch (IOException ioe) {System.out.println("IO error - initDB");
        }
        try {
            // load driver for appro
            Class.forName(driver);
        } catch (ClassNotFoundException cnfe) {System.err.println(cnfe);
        }
        try {
            con = DriverManager.getConnection(url, user, password);
        } catch (SQLException se) {System.err.println(se);
        }
    }
}
    
```

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import java.io.*;
import javax.swing.*;

public class Session extends JFrame implements ActionListener {
    private String subject, orderFileName, textFileName, dbParamName;
    private Connection con;
    private String[] queryText, nextQuerySpec;
    private StringTokenizer st;
    private JButton startButton, queryQuestions, queryArray;
    private JButton startButton = new JButton("Start");
    private Query nextQuery;
    private PrintWriter pw;
    private Container content;
    private JFrame temp = new JFrame("Database Query Study");

    public Session(String name) {
        super(name);
        Dimension screenSize = getToolkit().getScreenSize();
        orderFileName = "queryOrder.txt";
        textFileName = "queryText.txt";
        dbParamName = "dbParams.txt"; //file with driver on first line
        URL on 2nd, user on 3rd, password on 4th
        content=this.getContentPane();
        temp.setBounds(screenSize.width*48/100, 0, screenSize.width*52/100
            , screenSize.height);
        temp.setVisible(true);
        temp.setBackground(Color.black);
        temp.setBounds(662, 300, 300, 130);
    }

    public void actionPerformed(ActionEvent e) {
        queryIterator = queryArray.iterator();
        while (queryIterator.hasNext()) {
            this.display(queryIterator.next());
            CodeFind cf = new CodeFind(((String[])orderList.get(0))[1], codeL
                istener, pw);
        }

    public void initDB() {
        try {
            RandomAccessFile dbParams = new RandomAccessFile(dbParam
                sName, "r");
            driver = dbParams.readLine();
            url = dbParams.readLine();
            user = dbParams.readLine();
            password = dbParams.readLine();
            dbParams.close();
        } catch (IOException ioe) {System.out.println("IO error - initDB");
        }
        try {
            // load driver for appro
            Class.forName(driver);
        } catch (ClassNotFoundException cnfe) {System.err.println(cnfe);
        }
        try {
            con = DriverManager.getConnection(url, user, password);
        } catch (SQLException se) {System.err.println(se);
        }
    }
}
    
```

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import java.io.*;
import javax.swing.*;

public class Session extends JFrame implements ActionListener {
    private String subject, orderFileName, textFileName, dbParamName;
    private Connection con;
    private String[] queryText, nextQuerySpec;
    private StringTokenizer st;
    private JButton startButton, queryQuestions, queryArray;
    private JButton startButton = new JButton("Start");
    private Query nextQuery;
    private PrintWriter pw;
    private Container content;
    private JFrame temp = new JFrame("Database Query Study");

    public Session(String name) {
        super(name);
        Dimension screenSize = getToolkit().getScreenSize();
        orderFileName = "queryOrder.txt";
        textFileName = "queryText.txt";
        dbParamName = "dbParams.txt"; //file with driver on first line
        URL on 2nd, user on 3rd, password on 4th
        content=this.getContentPane();
        temp.setBounds(screenSize.width*48/100, 0, screenSize.width*52/100
            , screenSize.height);
        temp.setVisible(true);
        temp.setBackground(Color.black);
        temp.setBounds(662, 300, 300, 130);
    }

    public void actionPerformed(ActionEvent e) {
        queryIterator = queryArray.iterator();
        while (queryIterator.hasNext()) {
            this.display(queryIterator.next());
            CodeFind cf = new CodeFind(((String[])orderList.get(0))[1], codeL
                istener, pw);
        }

    public void initDB() {
        try {
            RandomAccessFile dbParams = new RandomAccessFile(dbParam
                sName, "r");
            driver = dbParams.readLine();
            url = dbParams.readLine();
            user = dbParams.readLine();
            password = dbParams.readLine();
            dbParams.close();
        } catch (IOException ioe) {System.out.println("IO error - initDB");
        }
        try {
            // load driver for appro
            Class.forName(driver);
        } catch (ClassNotFoundException cnfe) {System.err.println(cnfe);
        }
        try {
            con = DriverManager.getConnection(url, user, password);
        } catch (SQLException se) {System.err.println(se);
        }
    }
}
    
```



```

content.add(qText1);
content.add(Box.createVerticalGlue(1));

qText2 = new JLabel("<html><div style='float:right'>queryText [1] * </div>
</html></div></html>", JLabel.CENTER);
qText2.setBorder(BorderFactory.createEmptyBorder(0,20,0,20));
qText2.setAlignmentX(Component.CENTER_ALIGNMENT);
content.add(qText2);
content.add(Box.createVerticalGlue(1));

addConcept.addKeyListener(enterListener);
addConcept.addKeyListener(enterListener);
nextQuery.addKeyListener(this);
addConcept.addKeyListener(this);
addDonePanel.setLayout(new BorderLayout(addDonePanel, BorderLayout.X_AXIS));

addDonePanel.add(Box.createHorizontalGlue(1));
addConcept.add(Box.createHorizontalGlue(1));
addDonePanel.add(nextQuery);
addDonePanel.add(Box.createHorizontalGlue(1));
content.add(addDonePanel);
content.add(Box.createVerticalGlue(1));

JPanel incExLabel = new JPanel();
incExLabel.setLayout(new BorderLayout(incExLabel, BorderLayout.X_AXIS));
incExLabel.add(incConLabel);
incExLabel.add(exConLabel);
content.add(incExLabel);
content.add(Box.createRigidArea(new Dimension(0,10)));

JPanel numTitle = new JPanel(new GridLayout(0,4));
numTitle.add(new JLabel("Concept", JLabel.CENTER));
numTitle.add(new JLabel("Patients with Concept", JLabel.CENTER));
numTitle.add(new JLabel("Supplies with Concept", JLabel.CENTER));
numTitle.add(new JLabel("Patients with Concept", JLabel.CENTER));
content.add(numTitle);
content.add(Box.createVerticalGlue(1));

incConPanel.setLayout(new BorderLayout(incConPanel, BorderLayout.Y_AXIS));
incConPanel.add(bottomGlue);
exConPanel.setLayout(new BorderLayout(exConPanel, BorderLayout.Y_AXIS));
exConPanel.add(bottomGlue);
incConPanel.setBorder(BorderFactory.createCompoundBorder(
    .createEtchedBorder());
borderFactory.createEmptyBorder(0,20,0,20), BorderFactory
    .createEtchedBorder());
exConPanel.setBorder(BorderFactory.createCompoundBorder(
    .createEtchedBorder(1), BorderFactory.createEmptyBorder(0,20,0,20), BorderFactory
    .createEtchedBorder(1));
incConPanel.add(Box.createRigidArea(new Dimension(0,10)));
exConPanel.add(incConPanel);
incExPanel.add(exConPanel);
content.add(incExPanel);

resultPanel.add(currentCountLabel);
currentResult.setEditable(false);
resultPanel.add(currentResult);
content.add(resultPanel);
} //displayQueryBox

public void saveQuery(PrintWriter pw) {
    String incEx, medProb;
    ListQueryIds();
    pw.println("Query pids: " + Integer.toString(queryPids.size()));
    pw.println("concepts: " + Integer.toString(concepts.iterator().
        toArray().length));
}

```

```

// Michael Lieberman
// Query.java
// 11/12/02

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import java.io.*;
import java.sql.*;

public class Query extends JFrame implements ActionListener {
    public String subject, queryType, queryTypeFull;
    private HashSet queryPids = new HashSet();
    public Concept currentConcept;
    private Connection con;
    private Boolean conceptActive;
    private Dimension screenSize;
    private Container content;
    public JLabel incConLabel;
    public JLabel exConLabel;
    public JLabel incExLabel;
    public JLabel currentCountLabel;
    public JButton addConcept = new JButton("Add New Concept");
    public JButton editQuery = new JButton("Query Complete");
    public JArraylist concepts = new JPanel();
    public JPanel incConPanel = new JPanel();
    public JPanel exConPanel = new JPanel();
    public JTextField currentCountResult = new JTextField("0", 10);
    public JPanel addDonePanel = new JPanel();
    public JPanel incExPanel = new JPanel(new GridLayout(0,2));
    public Component bottomGlue = Box.createVerticalGlue();
    public Component topGlue = Box.createVerticalGlue();

    public Query (String subject, String queryType, String[] queryText, Conn
        action con) {
        super("Query #" + queryText[0] + queryType);
        this.subject = subject;
        this.queryType = queryType;
        this.queryText = queryText;
        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                this.dispose();
            }
        });
        concepts = new ArrayList();
        if (queryType.equals("S") || queryTypeFull = "color=M00X400>SNOMED"
            || queryTypeFull = "color=#800000>GEMS";
            conceptActive = false;
            screenSize = getToolkit().getScreenSize();
            displayQueryBox();
        }

        public void displayQueryBox() {
            content = this.getContentPane();
            content.setLayout(new BorderLayout(content, BorderLayout.Y_AXIS));
            content.add(Box.createRigidArea(new Dimension(0,20)));
            screenSize = screenSize.width*49/100, screenSize.height*4/10, screenSize
                .width*52/100, screenSize.height*6/10)-40);
            qText1 = new JLabel("<html><div style='float:right'>queryText: </div></div>
            </html></div></html>", JLabel.CENTER);
            qText1.setAlignmentX(Component.CENTER_ALIGNMENT);
        }
    }
}

```

```

Jan 15, 03 16:15 Query.java Page 4/5
requestFocus();
if (! (e.getSource() instanceof JFrame)) {
    concepts.add(currentConcept);
    updateQFids();
    removeButton.setMaxSize(new Dimension(100, 30));
    JPanel conceptPanel = new JPanel();
    conceptPanel.setSize(new Dimension(400, 50));
    conceptPanel.setLayout(new BorderLayout());
    conceptPanel.add(removeButton);
    conceptPanel.add(Box.createHorizontalStrut(10));
    conceptPanel.add(new JLabel(currentConcept.getName()));
    conceptPanel.add(Box.createHorizontalGlue());
    conceptPanel.add(new JLabel(Integer.toString(currentConcept.getConceptFids().size()));
    conceptPanel.setBorder(BorderFactory.createEmptyBorder(7, 20, 7, 20));
    removeButton.addActionListener(new RemoveListener() {
        if (currentConcept.getIncEx() == Concept.INCLUDE)
            incConPanel.remove(bottomGlue);
            incConPanel.add(conceptPanel);
            incConPanel.validate();
            incConPanel.validate();
            content.validate();
        } else {
            exConPanel.remove(bottomGlue);
            exConPanel.add(conceptPanel);
            exConPanel.validate();
            exConPanel.validate();
            incExPanel.validate();
            content.validate();
        }
    });
    public void windowClosing(WindowEvent e) {
        conceptActive = false;
        e.getWindow().setVisible(false);
    }
};
WindowAdapter queryPanelListener = new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        if (concepts.isEmpty())
            JOptionPane.showMessageDialog(content, "Please press
            'Query Complete' if finished.");
        "To restart query, simply remove all concepts.",
        "Error", JOptionPane.PLAIN_MESSAGE);
    } else
        JOptionPane.showMessageDialog(content, "Please enter at least
        one concept.",
        "Error", JOptionPane.PLAIN_MESSAGE);
};
KeyListener enterListener = new KeyAdapter() {
    public void keyPressed(KeyEvent k) {
        if (k.getKeyCode() == KeyEvent.VK_ENTER) ((Button) getFocusOwner()).doClick();
    }
};

```

```

Jan 15, 03 16:15 Query.java Page 3/5
currentConcept = (Concept) it.next();
if (currentConcept.getIncEx() == Concept.INCLUDE) incEx =
    "INC";
else incEx = "EXC";
if (currentConcept.getConceptType() == Concept.MED) medPro
    b = "MED";
else medProb = "PROB";
pw.println(" " + currentConcept.getName() + "\n" + incEx + "\n"
    + medProb + "\n");
pw.println("update: " + Integer.toString(currentConcept
    .getConceptFids().size()));
currentConcept.listCodes(pw);
pw.println();
} // saveQuery
private void updateQFids() {
    queryFids.clear();
    boolean firstSet = true;
    while (it.hasNext()) {
        Concept c = (Concept) it.next();
        if (currentConcept.getIncEx() == Concept.INCLUDE) {
            if (firstSet)
                queryFids.addAll(currentConcept.getConcept
                    Fids());
            firstSet = false;
        } else
            queryFids.removeAll(currentConcept.getConceptFid
                s());
        }
        it = concepts.iterator();
        while (it.hasNext()) {
            currentConcept = (Concept) it.next();
            if (currentConcept.getIncEx() == Concept.EXCLUDE) queryF
                ids.removeAll(currentConcept.getConceptFids());
            currentCountResult.setText(Integer.toString(queryFids.size()));
        }
    }
    public void actionPerformed(ActionEvent e) {
        if (!conceptActive)
            addConcept();
        else if (e.getSource() == addConcept) {
            conceptActive = true;
            currentConcept = new Concept(queryType, conceptL
                istener, cont);
        } else {
            if (e.getSource() == nextQuery) {
                if (!concepts.isEmpty())
                    conceptActive = true;
                    this.dispose();
            } else
                JOptionPane.showMessageDialog(this,
                    "Please enter at least one concept.",
                    "Error", JOptionPane.PLAIN_MESSAGE);
        }
    }
    else
        JOptionPane.showMessageDialog(this, "Please finish active concept befor
        e proceeding.",
        "Error", JOptionPane.PLAIN_MESSAGE);
};
WindowAdapter conceptListener = new WindowAdapter() {
    public void windowClosed(WindowEvent e) {

```

Jan 15, 03 16:15 **Query.java** Page 5/5

```

};

private class RemoveListener implements ActionListener {
    Concept rlConcept;
    JPanel rlPanel;

    public RemoveListener (Concept rlConcept, JPanel rlPanel) {
        this.rlConcept = rlConcept;
        this.rlPanel = rlPanel;
    }

    public void actionPerformed (ActionEvent event) {
        concepts.remove(rlConcept);
        updateQids();
        if (rlConcept.getIncEx() == Concept.INCLUDE) {
            incCompPanel.remove(rlPanel);
            incExpPanel.validate();
            content.validate();
            content.repaint();
        }
        else {
            exCompPanel.remove(rlPanel);
            exCompPanel.validate();
            incExpPanel.validate();
            content.validate();
            content.repaint();
        }
    } // actionPerformed
} // RemoveListener

public String getQueryNumber() {
    return queryText[0];
}

public String getQueryType() {
    return queryType;
}

private void listConcepts() {
    Iterator it = concepts.iterator();
    while (it.hasNext()) {
        System.out.println(((Concept)it.next()).getName());
    } //listConcepts
}

private void listQueryIds() {
    try {
        PrintWriter qw = new PrintWriter(
            new BufferedOutputStream(
                new FileOutputStream("g
                queryType+queryText[0]+".txt"));
        Iterator it = queryIds.iterator();
        while (it.hasNext()) {
            qw.println((String)it.next());
        }
        qw.flush();
        qw.close();
    } catch (IOException e) {System.out.println(e);}
} // listQueryIds
}

```

Wednesday April 30, 2003

Query.java



```

on con) { //Constructor
    public Concept(String queryType, WindowAdapter conceptListener, Connecti
        this.queryType = queryType;
        this.conceptListener = conceptListener;
        this.con = con;
        try {
            stmt = con.createStatement();
            screenSize = getToolkit().getScreenSize();
            initFrame.addWindowListener(conceptListener);
            initFrame.setName("initFrame");
            displayInitPanel();
        } //Constructor
        private void displayInitPanel() {
            top.setBorder(BorderFactory.createEmptyBorder(20, 20, 30, 30));
            top.add(nameLabel);
            inExRadioPanel.setLayout(new GridLayout(1, 2, 20, 20));
            inExRadioPanel.setBorder(BorderFactory.createCompoundBorder(
                createEmptyBorder(20, 20, 20, 20),
                BorderFactory.createLowerBevelBorder());
            inExRadioPanel.add(inRadioButton);
            inExGroup.add(inRadioButton);
            inExGroup.add(exRadioButton);
            typeRadioPanel.setLayout(new GridLayout(1, 2, 20, 20));
            typeRadioPanel.setBorder(BorderFactory.createCompoundBorder(
                createEmptyBorder(20, 20, 20, 20)),
                BorderFactory.createLowerBevelBorder());
            typeRadioPanel.add(proceedButton);
            typeRadioPanel.add(inRadioButton);
            typeGroup.add(inRadioButton);
            typeGroup.add(proceedButton);
            proceedButton.addActionListener(this);
            proceedPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20,
                20));
            initPanel.setLayout(new GridLayout(0, 1));
            initPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));
            initPanel.add(top);
            initPanel.add(inExRadioPanel);
            initPanel.add(typeRadioPanel);
            initPanel.add(inRadioButton);
            initPanel.add(exRadioButton);
            content = initPanel.getContentPane();
            content.add(initPanel);
            initFrame.setBounds(screenSize.width*48/100, screenSize.height*5
                //
                initFrame.setVisible(true);
            public void showConceptFrame() {
                String frameTitle;
                if (queryType.equals("S")) frameTitle = "SNOMED";
                else frameTitle = "GEMS";
                conceptFrame = new JFrame(frameTitle + " Codes for " + name);
                conceptFrame.addWindowListener(conceptListener);
                codeField.setMinimumSize(new Dimension(220, 30));
            }
        }
    }
}

```

```

// Michael Liberman
// Concept.java
// 11/12/02
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import java.io.*;
import javax.swing.JTextArea;
import java.net.DatagramSocket;
import java.net.DatagramPacket;

public class Concept extends JFrame implements ActionListener {
    private String name, queryType, conceptTypeString, currentCode;
    private Inl inEx, conceptType;
    private HashSet conceptPids=new HashSet(); // Set of all patient IDs w/
    th Concept;
    private Statement stmt;
    private ResultSet rs;
    private String sqlEx;
    private ArrayList conceptCodes = new ArrayList();
    private ArrayList snomedCodes = new ArrayList();
    private ArrayList expSetCodes = new ArrayList();
    private String snomedCode = null;
    private boolean valid;
    private Connection con;
    private WindowAdapter conceptListener;
    private Dimension screenSize;

    static int INCLUDE = 1;
    static int EXCLUDE = 0;
    static int MED = 0;
    static int PROB = 1;

    private JFrame initFrame = new JFrame("Concept Name and Type");
    private JLabel inExLabel = new JLabel("Please enter your label for this concept.");
    private JLabel typeLabel = new JLabel("Include or exclude subjects?");
    private JTextField nameField = new JTextField(20);
    private Container content;
    private JPanel initPanel = new JPanel();
    private ButtonGroup inExGroup = new ButtonGroup();
    private JRadioButton inExRadioPanel = new JRadioButton();
    private JRadioButton inExRadioButton = new JRadioButton("Include subjects with concept");
    private JRadioButton exRadioButton = new JRadioButton("Exclude subjects with concept");
    private ButtonGroup typeGroup = new ButtonGroup();
    private JPanel typeRadioPanel = new JPanel();
    private JRadioButton medButton = new JRadioButton("Medication");
    private JRadioButton probButton = new JRadioButton("Problem");
    private JPanel proceedPanel = new JPanel();
    private JButton proceedButton = new JButton("Proceed to entering code(s)");
    private JFrame conceptFrame;
    private JPanel enterCodePanelMain = new JPanel();
    private JPanel enterCodePanelH = new JPanel();
    private JPanel enterCodePanelV = new JPanel();
    private JTextField codeField = new JTextField(8);
    private JButton currentConceptLabel = new JLabel("Current Concept Codes:");
    private JButton currentConceptButton = new JButton("Add Entered Code to Concept");
    private JTextArea codeArea = new JTextArea(10, 40);
    private JButton conceptScroll = new JScrollPane(codeArea);
    private JButton conceptCompleteButton = new JButton("Concept Complete");
    private JPanel pasteButton;
    private JPanel getBottom = new JPanel(new BorderLayout());
}

```

```

Feb 26, 03 15:09                                     Concept.java                                     Page 4/6
Printed by root

3OptionPane.showMessageDialog (initFrame, "Please ch
oose include or exclude.", "Error", JOptionPane.PLAIN_MESSAGE);
else {
    if (incButton.isSelected()) incBox = INCLUDE;
    else incBox = EXCLUDE;
    if (medButton.isSelected()) %&#pound; probButton.isSelected()
        JOptionPane.showMessageDialog (initFrame,
            "Please choose Medication or Problem.",
            "Error", JOptionPane.PLAIN_MESSAGE
        );
    else {
        if (medButton.isSelected()) {
            conceptType = MED;
            conceptTypeString = "Medication";
        }
        else {
            conceptType = PROB;
            conceptTypeString = "Problem";
            initFrame.dispose ();
        }
    }
}

public void mapSnomedCode () {
    snomedCodes.addAll (conceptCodes);
    conceptCodes.clear ();
    Iterator sit = snomedCodes.iterator ();
    while (sit.hasNext ()) {
        expstCodes = expandSnomed ((String) sit.next ());
        Iterator it = expstCodes.iterator ();
        sqlText = "select concept_id from "+ Session.WBP+
            " where sit.concept_id in ("+(String) it.next ();
        while (it.hasNext ()) {
            sqlText = sqlText + ", "+ (String) it.next ();
        }
        sqlText = sqlText + ")*";
        System.out.println (sqlText);
        try {
            %&#pound; stmt = stmt.executeQuery (sqlText);
            %&#pound; rs = rs.next ();
            %&#pound; conceptCodes.add (rs.getString (1));
            %&#pound; catch (SQLException se) { System.err.println (se); }
        }
    }
}

// mapSnomedCode

public void selectFids () {
    Iterator it = conceptCodes.iterator ();
    if (it.hasNext ()) {
        sqlText = "select patient_key "+
            "from inqic_ "+ conceptTypeString + "_f
            = d_ "+ conceptTypeString + "_key and";
            if (conceptTypeString.equals ("problem")) sqlText = sqlText
            t +
            f, "bf, noc: 'tp') and";
            sqlText = sqlText + "d.concept_id in ("+(String) it.next ();
            while (it.hasNext ()) {
                sqlText = sqlText + ", "+ (String) it.next ();
            }
            sqlText = sqlText + ")*";
            System.out.println (sqlText);
            try {
                %&#pound; stmt = stmt.executeQuery (sqlText);
                %&#pound; rs = rs.next ();
                %&#pound; conceptFids.add (rs.getString (1));
                %&#pound; catch (SQLException se) { System.err.println (se); }
            }
        }
    }
}

```

```

Feb 26, 03 15:09                                     Concept.java                                     Page 3/6
Wednesday April 30, 2003

enterCodePanelH.setLayout (new BorderLayout (enterCodePanelH, BorderLayout
t.X_AXIS));
enterCodePanelH.add (new JLabel ("Enter "+ frameTitle + " code:"));
enterCodePanelH.add (Box.createGridArea (new Dimension (10, 0)));
enterCodePanelH.add (codeField);

t.Y_AXIS);
enterCodePanelV.setLayout (new BorderLayout (enterCodePanelV, BorderLayout
enterCodePanelV.add (enterCodePanelH);
enterCodePanelV.add (Box.createGridArea (new Dimension (0, 10)));
enterCodePanelV.add (enterCodeButton);

enterCodePanelMain.setLayout (new BorderLayout (enterCodePanelMain, B
pasteButton = new JButton ("<thumb><font face=Dialog size=12><font
FrameTitle = Code<br>From Clipboard</font><center><thumb>");
pasteButton.setPreferredSize (new Dimension (200, 70));
pasteButton.setMinimumSize (new Dimension (200, 70));
pasteButton.addActionListener (new PasteListener ());
enterCodePanelMain.add (pasteButton);
enterCodePanelMain.add (Box.createGridArea (new Dimension (20, 0)));

JLabel orLabel = new JLabel ("<thumb><font face=Dialog size=12><font<br>
orLabel.setPreferredSize (new Dimension (30, 30));
enterCodePanelMain.add (orLabel);
enterCodePanelMain.add (Box.createGridArea (new Dimension (12, 0)));

enterCodePanelMain.add (enterCodePanelV);
enterCodePanelMain.add (Box.createHorizontalGLue ());

enterCodeButton.addActionListener (new addCodeListener ());

codesArea.setEditable (false);

curConceptLabel.setAlignmentX (Component.CENTER_ALIGNMENT);
codesScroll.setPreferredSize (new Dimension (220, 100));
codesScroll.setAlignmentX (JPanel.CENTER_ALIGNMENT);

conceptCompleteButton.setAlignmentX (Component.CENTER_ALIGNMENT);
conceptCompleteButton.addActionListener (new conceptCompleteListener ());

content = conceptFrame.getContentPane ();
content.setLayout (new BorderLayout (content, BorderLayout.V_AXIS));
content.add (Box.createGridArea (new Dimension (0, 15)));
content.add (enterCodePanelMain);
content.add (Box.createGridArea (new Dimension (0, 12)));
content.add (curConceptLabel);
content.add (Box.createGridArea (new Dimension (0, 12)));
content.add (codesScroll);
content.add (Box.createGridArea (new Dimension (0, 10)));
content.add (Box.createGridArea (new Dimension (0, 10)));
conceptFrame.setBounds (screenSize.width/48/100, 0, screenSize.width
h*52/100, screenSize.height*4/10);
conceptFrame.setVisible (true);

}

public void validate () {
    if (nameField.getText (). equals (""))
        JOptionPane.showMessageDialog (initFrame, "Please enter a name fo
        r this concept.", "Error", JOptionPane.PLAIN_MESSAGE);
    else {
        name = nameField.getText ();
        if (incButton.isSelected ()) %&#pound; incButton.isSelected ()
    }
}

```

```

        codeField.setText("");
        conceptCodes.add(currentCode);
    }
    codeField.requestFocus();

    public boolean isEmpty() {
        if (queryType.equals("g")) return conceptCodes.isEmpty();
        else return false;
    }

    public String getName() {return name;}
    public int getInIdx() {return indEx;}
    public int getConceptType() {return conceptType;}
    public String getSnomedCode() {return "SNOMED CODE*";} //needs fixing
    public ArrayList getCodes() {return conceptCodes;}
    public HashSet getConceptIds() {return conceptIds;}
    public void listCodes(PrintWriter pw) {
        if (queryType.equals("g")) {
            iterator it = snomedCodes.iterator();
            pw.println("Unencod SNOMED Codes*");
            while (it.hasNext()) {
                pw.println("U"+(String)it.next());
            }
        }
        iterator it = conceptCodes.iterator();
        pw.println("UCENAS Codes*");
        while (it.hasNext()) {
            pw.println("U"+(String)it.next());
        }
        pw.flush();
    } //listCodes
}

```

```

private ArrayList expandSnomed(String sctMain) {
    ArrayList sctCodeList = new ArrayList();
    sctCodeList.add(sctMain);
    sqlText = "select sct_concept_id1 from sct_relationships+";
    relationship_type = 116680003u+";
    sqlText = "connect by sct_concept_id2 = prior sct_concept_id1 and r
ship_type = 116680003";
    System.out.println(sqlText);
    try {
        is = stmt.executeQuery(sqlText);
        while (rs.next()) {sctCodeList.add(rs.getString(1));}
    } catch (SQLException se) {System.err.println(se);}
    return sctCodeList;
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == addButton) validate();
    enterCodePanelMain.setText("");
    enterCodeButton.setAlignmentX(Component.CENTER_ALIGNMENT);
    if (!initFrame.isShowing()) showConceptFrame();
}

public class pasteListener implements ActionListener {
    public void actionPerformed(ActionEvent event) {
        Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();
        Transferable clipData = clipboard.getContents(this);
        String currentCode="";
        try {
            currentCode = (String)clipData.getTransferData(
DataFlavor.stringFlavor);
        } catch (Exception ex) {
            System.out.println(ex);
        }
        if (!currentCode.equals("")) {
            conceptCodes.add(currentCode);
            codeArea.append(currentCode+"u");
        }
    }
}

public class conComplexListener implements ActionListener {
    public void actionPerformed(ActionEvent event) {
        if (conceptCodes.isEmpty()) {
            JOptionPane.showMessageDialog(conceptFrame, "Plus
c entre a concept code.",
            "Error", JOptionPane.PLAIN_MESSAGE);
        }
        else {
            if (queryType.equals("g")) mapSnomedToCe();
            selectIds();
            conceptFrame.dispose();
            codeField.requestFocus();
        }
    }
}

public class addCodeListener implements ActionListener {
    public void actionPerformed(ActionEvent event) {
        String currentCode = codeField.getText();
        if (!currentCode.equals("")) {
            codeArea.append(currentCode+"u");
        }
    }
}

```

```

complete = true;
output = "";
while (it.hasNext()) {
    currentQuestion = (Question)it.next();
    if (currentQuestion.getResponse().equals("")) complete = false;
    else output = output + "uQuestion#" + it.nextIndex() + " :> " +
        currentQuestion.getResponse();
    if (complete) {
        pw.println(output);
        JOptionPane.showMessageDialog(this, "Session finished. Thanks"
            +
            _MESSAGE);
        this.dispose();
        pw.close();
        System.exit(0);
    }
    else JOptionPane.showMessageDialog(this, "Please answer all questions.",
        "Error", JOptionPane.PLAIN_MESSAGE);
}

private class Question extends JPanel implements ActionListener {
    JButton currentButton;
    ButtonGroup currentGroup = new ButtonGroup();
    JPanel answerPanel = new JPanel();
    String response = "";

    public Question (String text, int answersCount) {
        this.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createLoweredBevelBorder(0), BorderFactory.
            createEmptyBorder(20, 20, 20, 20));
        this.setLayout (new BorderLayout (this, BorderLayout.Y_AXIS));
        answerPanel.setLayout (new BorderLayout (answerPanel, BorderLayout.X_AXIS));
        answerPanel.add(Box.createVerticalGlue());
        for (int i=1; i<answersCount; i++) {
            currentButton = new JButton ("");
            answerPanel.add (currentButton);
            answerGroup.add (currentButton);
            currentButton.addActionListener (this);
        }
        answerPanel.setAlignmentX (Component.LEFT_ALIGNMENT);
        this.add (new JLabel (text, JLabel.CENTER));
        this.add (Box.createRigidiArea (new Dimension (0, 10)));
    }

    public void actionPerformed (ActionEvent event) {
        response = ((JRadioButton)event.getSource()).getText();
    }

    public String getResponse () {
        return response;
    }
}

```

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import java.io.*;
import java.net.*;
import java.sql.*;

public class Survey extends JFrame implements ActionListener {
    private JPanel mainPanel = new JPanel();
    private JLabel titleLabel = new JLabel ("<html><font face='Dialog' size=14><br><center><br></center></html>");
    private JLabel questionLabel = new JLabel ("Following questions '5' is the best and '1' is the worst<br><form></form></html>", JLabel.CENTER);
    private JButton submitButton = new JButton ("Submit");
    private PrintWriter pw;
    private ArrayList questionList = new ArrayList();
    private ArrayList answerList = new ArrayList();
    private Question currentQuestion;
    private JButton submitButton = new JButton ("Submit");
    private String output;
    private boolean complete;

    public Survey (PrintWriter pw) {
        this.pw = pw;
        setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        questionList.add ("Please rate ease of use of the SNOMED method (1 - very difficult, 5 - very easy)");
        questionList.add ("Please rate ease of use of the GEMS method (1 - very difficult, 5 - very easy)");
        questionList.add ("How likely would you be to use the SNOMED method (1 - very unlikely, 5 - very likely)");
        questionList.add ("How likely would you be to use the GEMS method (1 - very unlikely, 5 - very likely)");
        mainPanel.setLayout (new BorderLayout (mainPanel, BorderLayout.Y_AXIS));
        mainPanel.setBorder (BorderFactory.createEmptyBorder (20, 20, 20, 20));
        mainPanel.add (titleLabel);
        mainPanel.add (Box.createRigidArea (new Dimension (0, 20)));
        ListIterator it = questionList.listIterator();
        while (it.hasNext()) {
            currentQuestion = new Question (wrap ((it.nextIndex () - 1) + "<br><br>",
                questionList.add (currentQuestion);
            mainPanel.add (currentQuestion);
        }
        mainPanel.add (Box.createRigidArea (new Dimension (0, 15)));
        JPanel submitPanel = new JPanel ();
        submitPanel.setAlignmentX (JButton.CENTER);
        submitPanel.add (submitButton);
        mainPanel.add (submitButton);
        this.getContentPane ().add (mainPanel);
        this.setBounds (screenSize.width / 48 / 100, 0, screenSize.width * 52 / 100,
            screenSize.height);
        this.setVisible (true);

        private String wrap (String s) {
            return s.replaceAll ("<br>", "<br><br>");
        }

        public void actionPerformed (ActionEvent e) {
            ListIterator it = questionList.listIterator();

```

```

public void constructFrame() {
    setTitle("CodeFind");
    setSize(new Dimension(220, 300));
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JPanel mainPanel = new JPanel();
    mainPanel.setLayout(new BorderLayout());
    codeField.setText("");
    codeField.setAlignmentX(Component.CENTER_ALIGNMENT);
    codeTypeLabel.setText("Code Type");
    codeTypeLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
    enterLabel.setText("Enter Code");
    enterLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
    enterCodePanelMain.setLayout(new BorderLayout());
    enterCodePanelMain.add(enterLabel);
    enterCodePanelMain.add(Box.createVerticalGlue());
    enterCodePanelMain.add(Box.createRigidArea(new Dimension(0, 20)));
    enterCodePanelMain.add(enterCodeButton);
    enterCodePanelMain.add(Box.createVerticalGlue());
    content = this.getContentPane();
    content.add(enterCodePanelMain);
    updatePanel();
    this.setBounds(new Rectangle(0, 0, 220, 300));
    this.setVisible(true);
}

private void endQuiz() {
    JOptionPane.showMessageDialog(this, "Term lookup complete. Continue on to quiz.", "Terms Complete", JOptionPane.PLAIN_MESSAGE);
    this.dispose();
}

private void updatePanel() {
    codeTypeLabel.setText("Please enter the code");
    queryTypeString = "<font color=#006400><html currentTerm=" + currentTerm + "</font><center></html></font></center></html>";
    enterLabel.setText("Enter Code");
    content.validate();
}

public class ActionListener implements ActionListener {
    public void actionPerformed(ActionEvent event) {
        Transferable clipboard = clipboard.getContents(this);
        String clipCodes = "";
        try {
            clipCode = (String) clipboard.getTransferData(DataFlavor.stringFlavor);
        } catch (Exception ex) {
            System.out.println(ex);
        }
    }
}
    
```

```

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;

public class CodeFind extends JFrame implements ActionListener {
    private String name, queryType, queryTypeString, currentCode, currentTerm;
    private ArrayList termsQuestions;
    private Iterator quizIt;
    private WindowAdapter codeListeners;
    private Dimension screenSize;

    private JPanel enterCodePanelMain = new JPanel();
    private JTextField codeField = new JTextField(8);
    private JLabel codeTypeLabel = new JLabel("Code Type");
    private JButton enterCodeButton = new JButton("Submit");
    private JButton quitButton = new JButton("Quit");
    private JPanel questionPanel = new JPanel(new BorderLayout());
    private Container content;

    public CodeFind(String queryType, WindowAdapter codeListener, PrintWriter pw) {
        //Constructor "Term Lookup Exercise";
        this.name = queryType;
        this.codeListener = codeListener;
        this.pw = pw;
        this.setDefaultCloseOperation(WindowConstants.DO_NOTHING_ON_CLOSE);

        termsFileName = "terms.txt";
        screenSize = getToolkit().getScreenSize();
        this.addWindowListener(codeListener);
        loadTerms();
        if (queryType.equals("S")) queryTypeString = "SNOWMED";
        else queryTypeString = "GEMS";
        pw.println("Term lookup trial start at: " + System.currentTimeMillis() + " millis");

        resetQuiz();
        constructFrame();
    }

    private void resetQuiz() {
        currentTerm = (String) quizIt.next();
    }

    private void loadTerms() {
        termsQuestions = new ArrayList();
        try {
            RandomAccessFile termsFile = new RandomAccessFile(termsFileName, "r");
            String nextLine = termsFile.readLine();
            do {
                termsQuestions.add(nextLine);
            } while (nextLine != null);
            termsFile.close();
        } catch (IOException ioe) {
            System.out.println("IO error - loadTerm");
        }
    }
}
    
```

```

    }
    codeField.setText(clipCode);
}

public void actionPerformed(ActionEvent event) {
    String currentCode = codeField.getText();
    if (!currentCode.equals("")) {
        pw.println(queryTypeString + code for "currentTerm": "cur
            " entered at " + System.currentTimeMillis() + " milliseconds
        pw.flush();
        codeField.setText("");
        if (quizIt.hasNext()) {
            currentTerm = (String)quizIt.next();
        }
        else if (queryType.equals("S") && queryTypeString.equals
            queryTypeString = "GEMS";
            resetQuiz();
        }
        else if (queryType.equals("G") && queryTypeString.equals
            queryTypeString = "SNOMED";
            resetQuiz();
        }
        else endQuiz();
    }
    updatePanel();
    codeField.requestFocus();
}
}
}

```