SANDS: A SERVICE-ORIENTED ARCHITECTURE FOR CLINICAL DECISION SUPPORT IN A NATIONAL HEALTH INFORMATION NETWORK

by

Adam Wright

A DISSERTATION

Presented to the Department of Medical Informatics and Clinical Epidemiology

and the Oregon Health & Science University

School of Medicine

in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

May 2007

School of Medicine

Oregon Health & Science University

CERTIFICATE OF APPROVAL

This is to certify that the Ph.D. dissertation of

Adam Wright

has been approved

Advisor - Holly Jimison, PhD

Member - Dean F. Sittig, PhD

Member - Judy Logan, MD, MS

Member - Kent Spackman, MD, PhD

Member - John Halamka, MD, MS

Member - Blackford Middleton, MD, MPH, MSc

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

A research project of this magnitude could never have been completed without the help of many people, to each of whom I am deeply indebted.

First, I owe great thanks to my advisors and mentors Holly Jimison and Dean Sittig. I knew Drs. Sittig and Jimison long before I began my dissertation, and both of them have been instrumental not only in this project but in helping me begin my career in the field of medical informatics. They consistently provided leadership, encouragement, perspective and feedback throughout the process of writing my dissertation. Without their assistance, I could never have completed this project.

I also owe great thanks to Judy Logan and Kent Spackman, the other two members of my advisory committee. I asked Drs. Logan and Spackman to join my committee because I knew they would bring special expertise and perspective to the table, and they did not disappoint. Both of them faithfully attended my committee meetings, provided constructive feedback on my work, and helped me overcome the hurdles that naturally crop up during a research project. Thank you.

Next, I wish to thank John Halamka and Blackford Middleton, both on the faculty of the Harvard Medical School. Drs. Halamka and Middleton graciously served as outside members on my examination committee and brought a real world perspective to my work. Dr. Halamka is the chair of the Health Information Technology Standards

Panel (HITSP), whose efforts significantly influenced my work. Dr. Middleton is head of Clinical Informatics Research and Development (CIRD) at Partners Healthcare. I spent a summer at CIRD, where I developed the taxonomy presented in Chapter 3, as well as much of the service definition for the SANDS architecture.

The Department of Medical Informatics and Clinical Epidemiology has been my academic home for the past three years. The program has flourished under the leadership of William Hersh, and I was lucky to have the opportunity to train under Dr. Hersh and the rest of the department's faculty. The department is supported by a gifted administrative staff, including Andrea Ilg, Lynne Schwabe, Ashley Jones, Jeani Crichlow and Linda Slattery, all of whom graciously provided assistance and support to me throughout my project.

I had the opportunity to spend the summer of 2005 in the Office of the National Coordinator for Information Technology in the United States Department of Health and Human Services. It was there that my idea for the SANDS architecture germinated. I owe thanks to Dr. David Brailer, the first National Coordinator, as well as his staff, including Lori Evans, Lammot Dupont, Missy Krasner, Lewis Mattison, Richard Singerman, Kathleen Fyffe, Steve Posnack, Vish Sankaran and Katy Jensen, all of whom provided thoughtful and creative advice about the architecture.

I refined my architecture significantly over the course of my service on the steering committee for the Roadmap for National Action on Clinical Decision Support, and am very grateful to my colleagues from the steering committee: Jerry Osheroff, Jonathan Teich, Blackford Middleton, Elaine Steen and Don Detmer, as well as Richard Singerman and Karen Bell, who participated in the committee on an *ex officio* basis.

During my summer in CIRD I worked under Howard Goldberg and Barry Blumenfeld, who provided outstanding support and leadership. Marilyn Paterno, Vipul Kashyap and Tonya Hongsermeier, all of CIRD also provided significant feedback about my dissertation project.

While in Boston, I had the privilege of meeting Peter Szolovitz of MIT, Ken Mandel of Boston Children's Hospital, Henry Chueh of the Massachusetts General Hospital Laboratory of Computer Science, John Glaser of Partners HealthCare, Robert Greenes of the Brigham and Women's Hospital Decision Systems Group and Eric Poon and Tejal Gandhi, both from the Brigham and Women's Hospital. Each of them provided helpful advice, both about informatics careers and my research.

While at OHSU, I had the privilege of working with an outstanding group of National Library of Medicine Fellows. Each provided helpful feedback, and often acted as sounding boards for ideas relating to my architecture.

Over the past three years, I've been honored to participate as the student representative to the board of the Oregon Chapter of the Healthcare Information Management Systems Society. I thank my fellow board members for the opportunity to serve in this role, and also for the helpful real-world perspective they have shared with me as I conducted my research.

The OHSU Information Technology Group (ITG) graciously provided me a large number of surplus computers for use in my prototype and simulations. I am grateful to Bulisani Sibanda and Denise Ingram who coordinated the transfer of these computers. I am also grateful to the Clinical Outcomes Research Initiative who provided space in their server room for these computers.

The Markle Foundation, developers of the *Common Framework* for sharing health information has taken a recent interest in the architecture I have developed. David Lansky, Clay Shirky and Carol Diamond of the Markle Foundation provided helpful information, feedback and encouragement, and helped me see how my architecture fits into a broader context of quality-oriented health information exchange.

The prototype I developed integrates services from a variety of sources, including Computer Science Corporation's NHIN prototype. I am grateful to them for providing access to this prototype, and particularly to Andrew Wise of the Regenstrief Institute, who answered several questions I had about interfacing with it. Isabel Healthcare and Lexi-Comp provided free access to their decision support service which I appreciated greatly. I also appreciate the help of a number of vendors, who wish to remain anonymous, who responded to my request for information about their decision support capabilities.

I also wish to thank Mark Musen, head of Stanford Medical Informatics. Dr. Musen provided some of the early encouragement for me to enter the field, and has provided me with helpful advice throughout my time as a student in informatics.

Jonathan Leviss, my AMIA CIS working group mentor has also provided helpful advice about research and careers in the field. I am also grateful to BJ Fogg, whose lab I worked in as an undergraduate. Dr. Fogg encouraged me to think of myself as a researcher, and the research skills he taught me have served me well throughout my dissertation project. Misha Pavel has also provided me helpful mentorship and advice.

This work, along with all of my training was supported by National Library of Medicine Training Grant 1T15 LM009461. I thank the National Library of Medicine for its support, and Drs. Spackman and Hersh who served as Principal Investigators on the training grant.

And finally, I thank my parents. Without their love and support, none of this would have been possible.

ABSTRACT

Myriad studies have shown that clinical decision support can reduce medical errors and improve healthcare quality in both inpatient and ambulatory settings. However, only a small number of sites (generally academic medical centers and large integrated delivery networks) make significant use of the most advanced and effective decision support interventions. Community hospitals and independent healthcare providers generally make only limited use of decision support, in fact frequently opting to disable it entirely in their clinical systems. This lack of use stems from a variety of causes, ranging from technical to political to economic; however, perhaps the main cause is resources: academic medical centers and integrated delivery networks are more likely to have clinical decision support committees and in-house development resources.

The natural solution to closing this gap seems to be content sharing – having the successful sites share their content with the rest of the hospitals and providers. In fact, medical informatics has worked on a variety of approaches for sharing content, starting with Arden Syntax in 1989. However, to this day, none of these content sharing systems have seen significant adoption and many have never made it out of the lab.

In this dissertation, I introduce a new approach to sharing decision support content which leverages existing work towards developing a National Health Information Network (NHIN). I call this approach SANDS: a Service-oriented Architecture for NHIN Decision Support. Most approaches for sharing decision support

content involve developing a *lingua franca* for encoding clinical knowledge. However, because clinical knowledge is diverse and complex, and not always easy to represent in the form of if-then rules, such approaches necessarily constrain the scope and type of clinical knowledge which can be represented. SANDS, by contrast, defines a set of interfaces that a decision support service should make available, but leaves the choice of knowledge representation up to the implementer.

In addition to this interface-oriented design, SANDS allows knowledge to be distributed. With SANDS, instead of storing all knowledge in the electronic health record, it is made available over a network which many can contribute to. For example, a medical specialty society might release its guidelines over SANDS, while AHRQ would release evidence reports, and a commercial vendor might provide (for a fee) access to its drug information database. This frees each care provider from having to manage and maintain a complete body of decision support content – the medical equivalent of re-inventing the wheel.

The basis for SANDS is a functional taxonomy of clinical decision support, developed through analyzing the decision support content of a large integrated delivery network. From this framework, I developed a complete technical architecture and service definition. In turn, I built a prototype of the architecture. This prototype integrates a prototype NHIN with a variety of decision support systems, ranging from drug interaction checking to diagnostic decision support. The prototype provided a test-bed to study the utility, performance and relative advantages of the SANDS architecture.

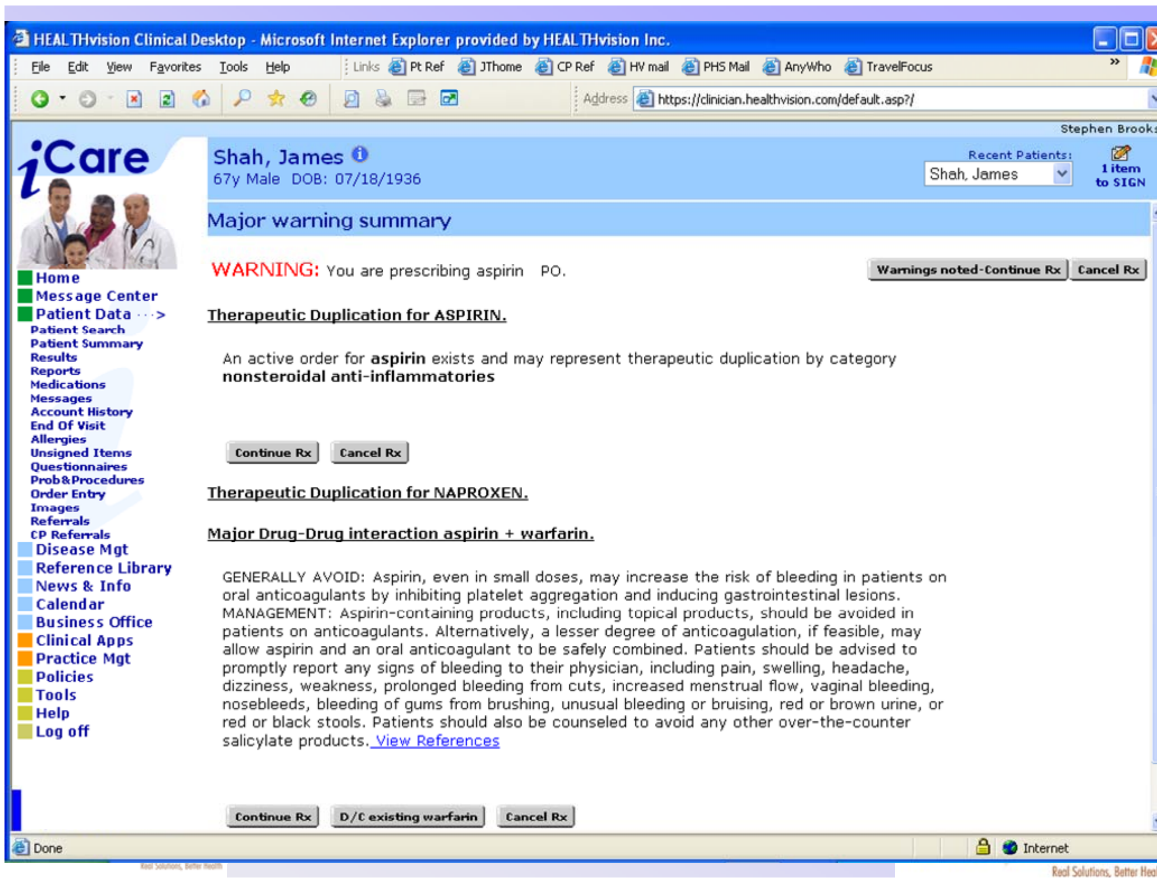
Ultimately, SANDS proved to be useful and effective, with good performance and significant technical and functional advantages over earlier approaches.

CHAPTER 1: BACKGROUND AND REVIEW OF THE LITERATURE

1.1 INTRODUCING CLINICAL DECISION SUPPORT

Clinical decision support has been defined in myriad ways, ranging from extremely precise and narrow definitions that exclude broad categories of work to extremely wide definitions that intentionally include efforts which might not be ordinarily thought of as clinical decision support. For the purposes of this dissertation a definition from the National Clinical Decision Support Task Force commissioned by the Office of the National Coordinator for Health Information Technology (ONC) in the US Department of Health and Human Services will be used. This definition defines clinical decision support as "Providing clinicians, patients or individuals with knowledge and person-specific or population information, intelligently filtered or presented at appropriate times, to foster better health processes, better individual patient care, and better population health."(1)

That said, it is instructive to look at a few examples of clinical decision support systems to gather a feeling for their breadth and for a few of the many sorts of applications for which such systems can be used. Like the definition, these examples come from the Roadmap for National Action on Clinical Decision Support and were assembled by the aforementioned task force. Figure 1 shows an example of a drug-drug interaction warning. In this example, a physician has prescribed aspirin to a patient who is also receiving naproxen, which therapeutically duplicates aspirin, and warfarin, which, when combined with aspirin, poses a significant risk for bleeding.



Shah, James
67y Male DOB: 07/18/1936

Major warning summary

WARNING: You are prescribing aspirin PO.

Therapeutic Duplication for ASPIRIN.

An active order for **aspirin** exists and may represent therapeutic duplication by category **nonsteroidal anti-inflammatories**

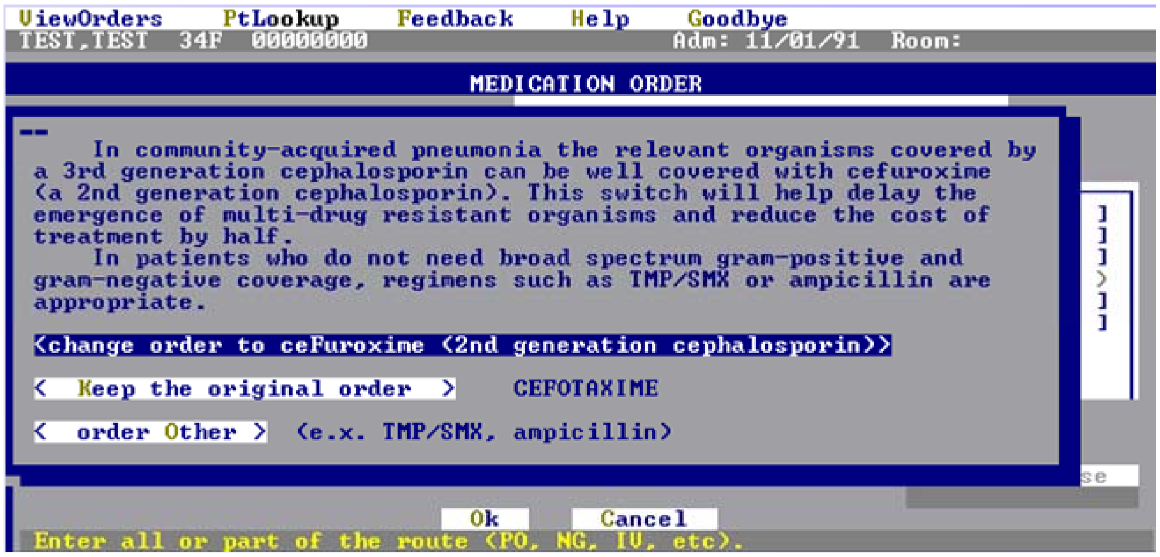
Therapeutic Duplication for NAPROXEN.

Major Drug-Drug interaction aspirin + warfarin.

GENERALLY AVOID: Aspirin, even in small doses, may increase the risk of bleeding in patients on oral anticoagulants by inhibiting platelet aggregation and inducing gastrointestinal lesions. MANAGEMENT: Aspirin-containing products, including topical products, should be avoided in patients on anticoagulants. Alternatively, a lesser degree of anticoagulation, if feasible, may allow aspirin and an oral anticoagulant to be safely combined. Patients should be advised to promptly report any signs of bleeding to their physician, including pain, swelling, headache, dizziness, weakness, prolonged bleeding from cuts, increased menstrual flow, vaginal bleeding, nosebleeds, bleeding of gums from brushing, unusual bleeding or bruising, red or brown urine, or red or black stools. Patients should also be counseled to avoid any other over-the-counter salicylate products. View References

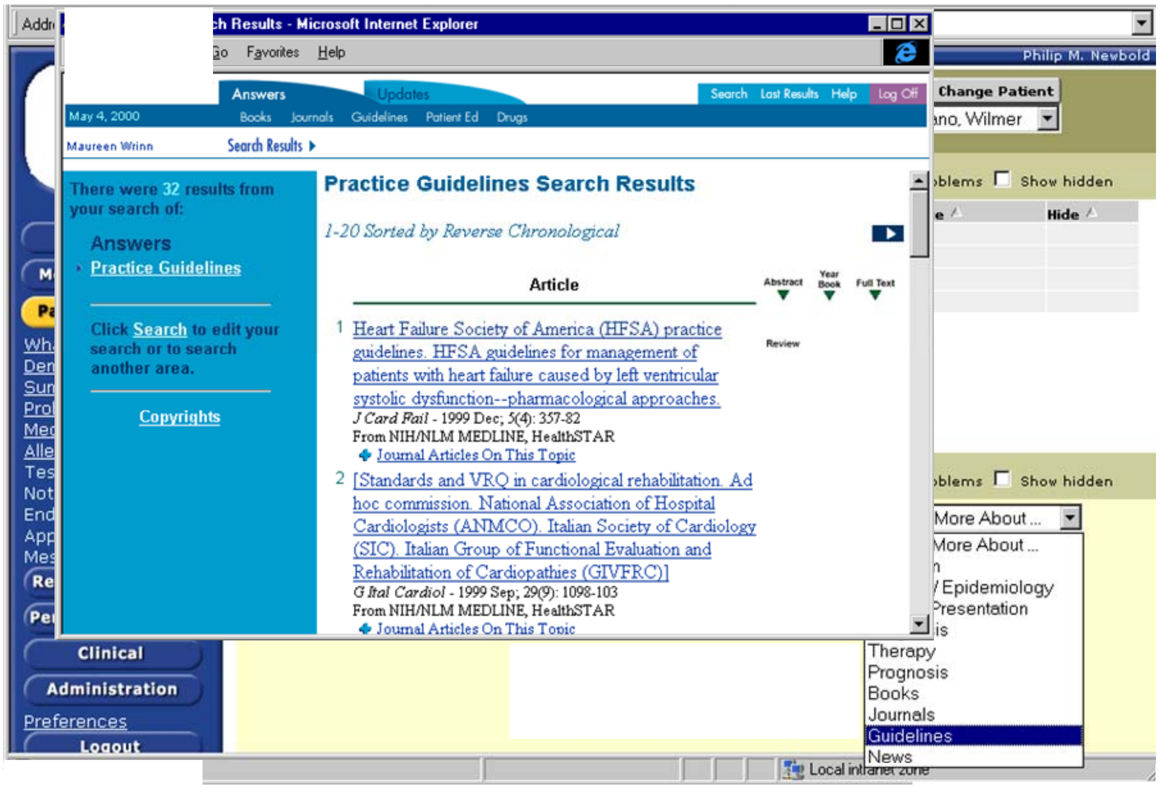
FIGURE 1 – A THERAPEUTIC DUPLICATION ALERT.

Figure 2 shows a screenshot of the Brigham Integrated Computing System (BICS) (2, 3) in use at the Brigham and Women's Hospital in Boston, MA. In this example, a physician has prescribed Cefotaxime, a third-generation cephalosporin for a patient with community acquired pneumonia. The computer has determined that, given the diagnosis, a second generation cephalosporin, such as Cefuroxime, would be just as effective but would cost less, have fewer side effects, and be less likely to contribute to antibiotic resistance.



```
ViewOrders     PtLookup     Feedback     Help     Goodbye
TEST,TEST   34F   00000000                    Adm: 11/01/91   Room:

                         MEDICATION ORDER
--
       In community-acquired pneumonia the relevant organisms covered by
   a 3rd generation cephalosporin can be well covered with cefuroxime
   (a 2nd generation cephalosporin). This switch will help delay the
   emergence of multi-drug resistant organisms and reduce the cost of
   treatment by half.
       In patients who do not need broad spectrum gram-positive and
   gram-negative coverage, regimens such as TMP/SMX or ampicillin are
   appropriate.

   <change order to ceFuroxime (2nd generation cephalosporin)>

   <  Keep the original order  >    CEFOTAXIME

   <  order Other >  (e.x. TMP/SMX, ampicillin)

                          Ok       Cancel
 Enter all or part of the route (PO, NG, IV, etc).
```

FIGURE 2 - A BEST PRACTICE ALERT IN THE BICS SYSTEM.



h Results - Microsoft Internet Explorer

May 4, 2000

Maureen Wrinn

Search Results

There were 32 results from your search of:

Answers

Practice Guidelines

Click Search to edit your search or to search another area.

Copyrights

Practice Guidelines Search Results

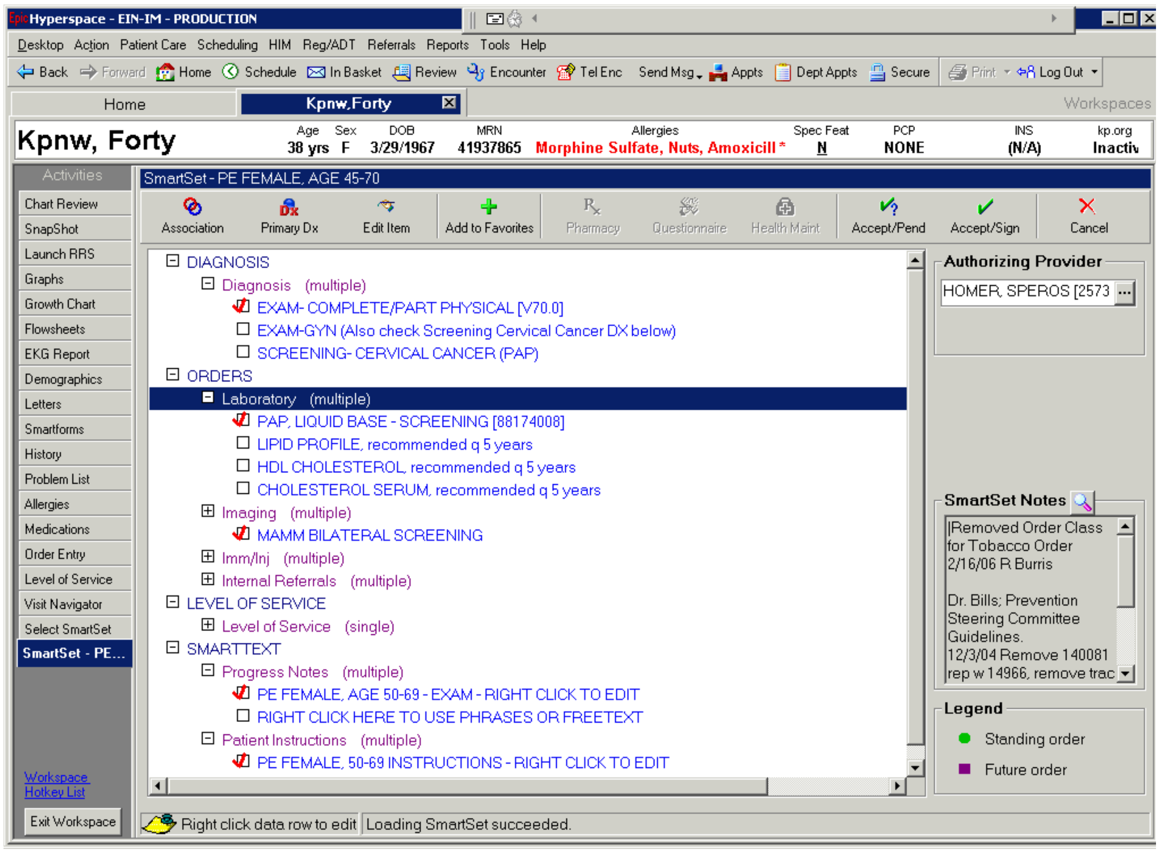
1-20 Sorted by Reverse Chronological

	Article	Abstract	Year Book	Full Text
1	Heart Failure Society of America (HFSA) practice guidelines. HFSA guidelines for management of patients with heart failure caused by left ventricular systolic dysfunction--pharmacological approaches. *J Card Fail* - 1999 Dec; 5(4): 357-82 From NIH/NLM MEDLINE, HealthSTAR Journal Articles On This Topic	Review		
2	[Standards and VRQ in cardiological rehabilitation. Ad hoc commission. National Association of Hospital Cardiologists (ANMCO). Italian Society of Cardiology (SIC). Italian Group of Functional Evaluation and Rehabilitation of Cardiopathies (GIVFRC)] *G Ital Cardiol* - 1999 Sep; 29(9): 1098-103 From NIH/NLM MEDLINE, HealthSTAR Journal Articles On This Topic			

FIGURE 3 - AN INFORMATION INTERVENTION.

Figure 3 shows the result of an Infobutton(4-6) query. In this case, the physician has entered a diagnosis of Congestive Heart Failure in a clinical system, and has executed a query via an Infobutton for clinical practice guidelines relating to heart failure.

Clinical decision support has also been used frequently for best practices and preventive care. Figure 4 shows a set of preventive care orders in the EpicCare system (Epic Systems Corporation, Madison, WI) as implemented at Kaiser Permanente in the Northwest(7, 8).


Hyperspace - EIN-IM - PRODUCTION
Desktop Action Patient Care Scheduling HIM Reg/ADT Referrals Reports Tools Help
Back Forward Home Schedule In Basket Review Encounter Tel Enc Send Msg Appts Dept Appts Secure Print Log Out
Home Kpnw,Forty Workspaces
Activities
Chart Review
SnapShot
Launch RRS
Graphs
Growth Chart
Flowsheets
EKG Report
Demographics
Letters
Smartforms
History
Problem List
Allergies
Medications
Order Entry
Level of Service
Visit Navigator
Select SmartSet
SmartSet - PE...
Workspace Hotkey List
Exit Workspace
Association Primary Dx Edit Item Add to Favorites Pharmacy Questionnaire Health Maint Accept/Pend Accept/Sign Cancel

Kpnw, Forty

Age	Sex	DOB	MRN	Allergies	Spec Feat	PCP	INS	kp.org
38 yrs	F	3/29/1967	41937865	Morphine Sulfate, Nuts, Amoxicill *	N	NONE	(N/A)	Inactiv

SmartSet - PE FEMALE, AGE 45-70

- DIAGNOSIS
  - Diagnosis (multiple)
    - ✓ EXAM- COMPLETE/PART PHYSICAL [V70.0]
    - ☐ EXAM-GYN (Also check Screening Cervical Cancer DX below)
    - ☐ SCREENING- CERVICAL CANCER (PAP)
- ORDERS
  - Laboratory (multiple)
    - ✓ PAP, LIQUID BASE - SCREENING [88174008]
    - ☐ LIPID PROFILE, recommended q 5 years
    - ☐ HDL CHOLESTEROL, recommended q 5 years
    - ☐ CHOLESTEROL SERUM, recommended q 5 years
  - Imaging (multiple)
    - ✓ MAMM BILATERAL SCREENING
  - Imm/Inj (multiple)
  - Internal Referrals (multiple)
- LEVEL OF SERVICE
  - Level of Service (single)
- SMARTTEXT
  - Progress Notes (multiple)
    - ✓ PE FEMALE, AGE 50-69 - EXAM - RIGHT CLICK TO EDIT
    - ☐ RIGHT CLICK HERE TO USE PHRASES OR FREETEXT
  - Patient Instructions (multiple)
    - ✓ PE FEMALE, 50-69 INSTRUCTIONS - RIGHT CLICK TO EDIT

Authorizing Provider
HOMER, SPEROS [2573

SmartSet Notes
|Removed Order Class for Tobacco Order 2/16/06 R Burris

Dr. Bills; Prevention Steering Committee Guidelines.
12/3/04 Remove 140081 rep w 14966, remove trac

Legend
● Standing order
■ Future order

Right click data row to edit | Loading SmartSet succeeded.

FIGURE 4 - A PREVENTIVE CARE ORDERSET.

The CDS Implementer's Guide (9) lays out a taxonomy of six types of clinical decision support systems:

1. Documentation forms / templates
2. Relevant data display
3. Order creation facilitators
4. Time-based checking and protocol/pathway support
5. Reference information and guidance
6. Reactive alerts and reminders

Other types of decision support, such as diagnostic decision support systems (10-18), fall outside of this taxonomy but are still important in the history and evolution of decision support.

The reasons for using decision support systems are numerous. Such systems have been used to prevent errors, improve quality, reduce costs and save time. The best evidence suggests that such systems, when used, can be extremely effective(19-25). For example, a recent systemic review by Garg found that in 100 studies of clinical decision support, "CDSS improved practitioner performance in 62 (64%) of the 97 studies assessing this outcome, including 4 (40%) of 10 diagnostic systems, 16 (76%) of 21 reminder systems, 23 (62%) of 37 disease management systems, and 19 (66%) of 29 drug-dosing or prescribing systems". (21)

Another recent review by Kawamoto and Lobach(25) identified four critical success factors for decision support systems: first, that such systems should automatically provide decision support without first requiring an intentional request by a clinician; second, that the system provide recommendations, not just assessments; third, that the system provides support at the "time and location of decision making"; and fourth, that the system be computer-based (paper-based and mechanical decision support systems are also sometimes used). In their analysis, of the systems that met all four of these criteria, 94% improved clinical practice significantly. While conducting this review of the literature it was not uncommon to see reports of a single system, operating at a single hospital and covering a single domain of medicine, which resulted in reduced mortality of, say, half a dozen lives per year. When spread across all the hospitals in the United States, and all the clinical domains in which a decision support system could be implemented, it is quite possible to see the potential of such systems to significantly reduce the 98,000 lives lost every year due to medical errors,(26) and to improve the quality of care adults in the United States receive – current estimates suggest that adults today receive only 54.9% of all care that evidence suggests they should.(27)

Despite their great potential, and a history of successes, clinical decision support systems have not found wide use outside of a handful of mostly academic medical centers, the VA, and large integrated delivery systems, such as Kaiser Permanente.(28) To understand why, it's necessary to look at the history and evolution of clinical decision support which is accomplished in the next section.

1.2 STANDALONE DECISION SUPPORT SYSTEMS

Pinpointing the precise beginning of the field of medical informatics is, of course, challenging but it is perhaps a safe choice to begin any discussion of the history of the field with the 1959 paper "Reasoning foundations of medical diagnosis; symbolic logic, probability, and value theory aid our understanding of how physicians reason" by Robert Ledley and Lee Lusted.(29) Robert Ledley went on to invent the whole-body CT scanner (30), and Lee Lusted became a leader in the field of medical decision making (31). Their 1959 paper, however, laid out a probabilistic model for medical diagnosis with grounds in set-theory and Bayesian inference. The article, published in *Science*, was, in many ways, a tutorial in statistical and probabilistic inference for clinicians. Amongst other contributions, the paper proposed an analog computer used to sort cards. These cards would contain a diagnosis and a series of punches which represented symptoms. By selecting the cards which matched the symptoms present in a given case a clinician could begin to develop a differential diagnosis. The number of cards supporting a particular diagnosis served as a measure of the likelihood of that diagnosis. The system could easily be updated as new patients were seen – the clinician simply had to fill out and punch a card for each new patient and diagnosis and drop it into the sorter.

Two years after Ledley and Lusted published their paper, Homer Warner of the University of Utah published a mathematical model for diagnosing congenital heart defects.(32) This model used contingency tables to map symptoms and signs to diagnoses based on the frequency of manifestation for each symptom or sign given an underlying diagnosis. The system was evaluated by comparing its diagnoses with gold-standard surgical diagnoses, and was found to compare favorably with experienced cardiologists.

Shortly after Warner, Morris Collen developed a system for "Automated Multiphasic Screening And Diagnosis"(33) which was used at Kaiser Permanente. When a patient came in for an exam he or she was given a stack of cards each of which contained a symptom or a question. The patient would put the cards which indicated a symptom he or she was experiencing, or a question to which their answer was affirmative, into a designated "Yes" box, and the rest of the cards into a "No" box. These cards were then run through a computer system which proposed an initial differential diagnosis.

In 1969 Howard Bleich developed a system to suggest therapy for acid-base disorders.(34) This system was unique because it was the first system to suggest a therapy in addition to a diagnosis. The system would take in information useful for diagnosing acid-base disorders, such as the results of arterial blood gas tests, vital signs, and clinical findings. If any information needed for decision-making was missing, the system would prompt the user to collect and enter that information. If the information was complete the system would produce an evaluation note, written in the same style as a human specialist consultant, proposing a management plan for review by the physician providing care.

Two years later, in 1971, de Dombal built a probabilistic model and computer system for diagnosing abdominal complaints.(11, 35) The system was significant because evaluation showed it to be quite effective. When compared to the final gold-standard surgical diagnosis, the computer's preliminary diagnosis was accurate 91.8% of the time. By comparison, a group of senior clinicians was correct 79.6% of the time. Not only was the computer able to match the performance of senior clinicians, it actually improved significantly on it – cutting the error rate in half.

In the 1970s, the field of artificial intelligence began to influence medical informatics in a significant way. In 1975 Shortliffe applied the tools and techniques of expert systems to the problem of antibiotic prescribing with his system, MYCIN.(36) MYCIN operated in three modes:

1. A consultation system which collected information, applied rules and recommended therapy.
2. An explainer system which explains its recommendations in English.
3. A rule acquisition system, used by an expert to build rules for the knowledge base.

Early evaluation showed it suggested acceptable therapy 75% of the time, but it got better as more rules were added.

Most of these early systems would, broadly, take input of clinical parameters and make suggestions of diagnoses or therapy. The ATTENDING system(37, 38) by Perry

Miller of Yale, however, took a different approach. The user of ATTENDING would, as with the other systems, input clinical parameters, but he or she would also enter a proposed plan. The system would then make comments and suggestions about the plan, and it would be up to the user to change the plan based on these suggestions. This method of interaction was called critiquing, and critiquing systems were eventually developed for ventilator management, hypertension, and other clinical domains.

The systems discussed heretofore all have one thing in common: they are limited to one specific area of medicine, such as antibiotic prescribing, or congenital heart defects. The INTERNIST-I system,(16, 39) developed by Randy Miller, Harry Pople and Jack Myers, on the other hand, attempted to provide diagnostic decision support across the entire field of internal medicine. The system's knowledge base, once fully developed, comprised "15 person-years of work, 500 disease profiles, 3550 manifestations of disease."(16) It was tested on 19 standardized clinical exercises published in the *New England Journal of Medicine*, and did about as well as an average doctor in proposing the correct diagnosis for the case, but not as well as the experts who wrote the cases up. One key intellectual contribution of the INTERNIST system was the way it abstracted the complex field of diagnosis into three concepts: evoking strength, frequency and import. Shortly after INTERNIST, Octo Barnett, and others, released the DXplain system.(10) The DXplain system is still available today, and has been updated with a web-interface. INTERIST eventually evolved into the QMR system, though neither system is still available.

All of the systems reviewed in this section fall into the category of standalone decision support systems. That is, they were systems (usually computer programs) which ran separately from any other system. To employ one, a clinician had to intentionally and purposefully seek the system out, log into it, enter information about his or her case, and then read and interpret the results. Such systems have several key advantages over other types of clinical decision support systems. First, anyone can make one. Doing so requires no special access to clinical data or clinical systems. There is no need to standardize on anything: completely arbitrary systems can be used for terminology, input structure, output format and knowledge representation. Such systems are also easy to share – the creator could simply mail the tapes to anyone else who wished to use his or her system (or, in the modern analog, upload the system to a website). Along with these advantages, though, standalone clinical decision support systems have some significant, and potentially disqualifying, disadvantages. First and foremost, people have to seek the systems out, so the system can't be proactive. But the cause of many medical errors is lack of knowledge – people don't know what they don't know. A system which isn't proactive cannot be of assistance in such a case. The other disadvantage is more practical – these systems were extremely inefficient to use. It could take well over an hour to enter a case into the INTERNIST system, so its use was, naturally, quite narrow. This limitation was especially bothersome where systems were dealing with data, such as lab results, that were likely available in electronic form in another system, but which still had to be manually entered due to a lack of integration.

1.3 DECISION SUPPORT INTEGRATED INTO CLINICAL SYSTEMS

To surmount the significant problems with standalone clinical decision support systems, developers began integrating such systems into other clinical systems, such as computerized physician order entry (CPOE) and electronic health record (EHR) systems. The HELP system(40), developed at the University of Utah, and used at LDS Hospital in Salt Lake City was the first example of such integration. HELP had advanced Bayesian decision support capabilities, and included modules for a variety of functions, including many of the functions described above. HELP also served as the substrate for many successful student projects in clinical decision support. One particularly notable project was the COMPAS system(41, 42) developed by Dean Sittig. COMPAS was developed as part of a trial of $ECCO_2$, a new ventilation modality for patients suffering from acute respiratory distress syndrome (ARDS). The ventilation protocol was so complex that no human could fully carry it out, so COMPAS was developed to provide computerized execution of the protocol. It used the Blackboard Control Architecture, developed at Stanford, to update and manage its internal state. The system interfaced with a variety of other systems, and used data ranging from lab values, past ventilator settings and clinical observations to recommend ventilator changes. The critically ill patients in the trial had a 41% survival rate, compared to an historical rate of only 9%. The HELP system also served as the basis for a critiquing system for blood ordering developed by

Reed Gardner(43) and a well-known antibiotic advising system developed by Scott Evans.(44)

Around the same time as the HELP system, Clem McDonald of the Regenstrief Institute in Indiana was developing the Regenstrief Medical Record System (RMRS). This system used a large base of rules to make suggestions about care. The system was evaluated in an experimental trial. In the first half of the trial, half of the users of the RMRS system received suggestions based on the database of rules while half received no suggestions. Dr. McDonald found that physicians carried out the suggestions they received 51% of the time. However, when they didn't receive suggestions, they carried out the care only 22% of the time.(45) While this finding was, itself, quite important, the most significant finding came after all the alerts were turned off. Almost immediately, performance went back to baseline – the physicians who had been receiving the alerts, and providing higher quality care, dropped down to the level they had been at before the trial began. In other words, there was no learning effect. In a landmark paper, McDonald argued that medicine had become so complex, and the amount of information required to practice it effectively so expansive, that no un-aided human actor could provide perfect care. Instead, he contended, some sort of ancillary aid, most likely a computer, was needed.

In addition to HELP and RMRS, a variety of other clinical systems, mostly at academic medical centers, have been used for clinical decision support. The WizOrder system,(46-50) in use at Vanderbilt, and now available commercially as Horizon Expert

Orders from McKesson has been a fruitful development platform, as has BICS(2, 3, 51-57). The Veterans Health Administration has also been a leader in the field of clinical decision support with their Computerized Patient Record System (CPRS)(58, 59), reporting spectacular outcomes for even simple interventions, and quickly rocketing from a position as one of the lower-performing healthcare systems, to a quality superstar.

Integrating CDS into clinical information systems solved some problems, while creating others. Integrated systems have two key advantages over standalone systems: first, since they're integrated, the user doesn't have to reenter information which is already stored electronically; and, second, such systems can be proactive – they can alert a user to a dangerous drug-drug interaction or a dosing error without the user going out and seeking assistance. The major downside of integrated systems is that there is no way to share them or reuse their content. Standalone systems can be easily shared – often by simply mailing a tape or emailing an executable program. In contrast, since integrated systems are built directly into larger clinical systems they can't be directly shared with others who aren't using the same clinical system. Also, integrating decision support into a clinical system can create knowledge-management problems. It's hard to separate knowledge from code – if a clinical guideline is updated, it may be necessary to review the entire source code for a clinical system to find the places where this guideline is used. And finally, nearly everyone who has been successful at developing integrated decision support systems has also developed their own clinical system. However, the vast majority of hospitals and doctors buy rather than build clinical systems so this limitation has kept decision support from seeing wide adoption.

1.4 STANDARDS FOR SHARING DECISION SUPPORT CONTENT

In response to the inability to share decision support content, a variety of efforts have been undertaken to standardize clinical decision support content. Foremost among these is the Arden Syntax(60-64). The initial version of the Arden Syntax was developed at a three day consensus meeting in June of 1989 held at the Arden Homestead in New York from which the standard got its name. The standard combined the syntaxes used by the HELP system and the RMRS system because these systems were (and to an extent, still are) the two most prominent clinical systems. Both systems are discussed above. Rules encoded in Arden Syntax are called Medical Logic Modules. The Arden Syntax divides rules into three sections, called the "maintenance", "library" and "knowledge" sections. The maintenance section contains meta-data about the rule, such as who owns it, when it was created, when it was last reviewed or updated, and its validation status. The library section contains meta-data describing the clinical role of the rule, its purpose, an explanation, keywords, and a citation to the original source of the guideline or best practice that the rule encodes. The computable portion of the rule is encoded in the knowledge section. The knowledge section contains subsections called "type", "data", "evoke", "logic", "action" and "urgency". In the current version of Arden Syntax, type is always set to "data-driven" because this is the only mode of decision support offered. The data section is used to read data values, such as recent lab tests, medications lists, or clinical problems from the encompassing clinical system. The evoke section contains one or more triggers that might cause the rule to fire, such as "new potassium value stored". The logic section encodes the rule, generally as a series of if-then statements, and the action section encodes what the rule does when its logic section is satisfied – in general, Arden Syntax has only been used to raise alerts. The urgency section contains a number between 1 and 100 to encode how important that rule is. The guidelines used to assign urgencies is implementation dependent, and not fully defined in the specification. The Arden Syntax has had some limited commercial success. Three clinical system vendors (Eclipsys, McKesson and Siemens), which represent about a quarter of the overall clinical system market, offer some support for the Arden Syntax, and a number of vendors, most notably Thomson Micromedex and Zynx sell Medical Logic Modules. Arden syntax has two key limitations: first, it can only be used to encode event-driven, patient-specific rules. For use cases such as drug-drug interaction checking, or panic lab value alerting, this modality is sufficient. However, because Arden Syntax is patient-specific, it cannot be used for population-based decision support (such as a quality-of-care dashboard), and because it is event-driven, it can't be used for point-of-care reference or information retrieval support. The other key limitation relates to vocabulary: Arden Syntax does not define a standard vocabulary for things like lab tests, drugs or procedures. As a result, even if two clinical systems support the Arden Syntax, if they use different terminologies, Arden Syntax rules from one system cannot be used in the other system without modification. For example, if one hospital's clinical system stored a blood test result as "Serum Potassium" and another hospital's clinical system stored the same result as "K+" a human-guided mapping would be needed. To assist in this mapping, Arden Syntax wraps system-specific terminological expressions in curly braces, and automated tools exist to help the implementer disambiguate these bracketed terms, but human intervention is still required. This problem is so limited and well-known that it is referred to simply as the "curly braces problem." The Arden Syntax has been revised several times since it was first created in 1989, and its second version has been accepted as a standard by both the American National Standards Institute (ANSI), and Health Level 7 (HL7), a healthcare standards body.

Since the creation of Arden Syntax, numerous other standards for representing and sharing decision support content and knowledge have been created. Many of these efforts have stalled, but one effort in particular, the Guideline Interchange Format(65-69) (GLIF) has gained limited traction. GLIF was developed at Harvard by Lucila Ohno-Machado. Unlike Arden Syntax, which is mostly designed for alerts and reminders, GLIF focuses on more complex multi-part guidelines, though simpler Arden Syntax type rules can also be encoded in GLIF. GLIF takes a three-level approach to knowledge representation. The top level, termed the "conceptual" level, gives a high-level overview of the logic in a guideline, represented as a flow-chart. Figure 5 shows the conceptual-level representation of a guideline in GLIF and is taken from one of the first papers to describe the GLIF approach.(66) This conceptual level encoding is further refined into what GLIF calls a "computable" form – a formal representation of the exact logic of the rule.

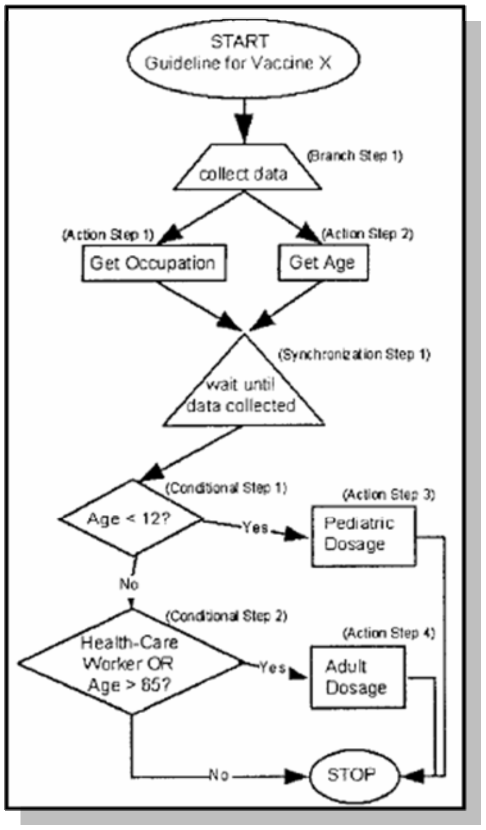


FIGURE 5 - CONCEPTUAL LEVEL REPRESENTATION OF AN INFLUENZA VACCINE GUIDELINE ENCODED IN GLIF.(66)

Like Arden Syntax, GLIF gives implementers working at the computable level wide terminological berth. Computable level guidelines, though formally described, are not designed to be directly executed in a clinical information system. Instead, GLIF envisions that these computable level guidelines will be translated *by a human* to executable code, termed the "implementable" level. This is a significant hurdle to the wide adoption of GLIF, because it means that GLIF guidelines cannot be directly loaded into a clinical system and executed – instead, they exist as a formal representation for an implementer to design a clinical system specific guideline. In later work, a general-purpose execution engine for computable level guidelines was described(69), but it has not been implemented in any commercially available system. Like Arden Syntax, GLIF has been revised several times – the current version is GLIF3.

Although GLIF is not currently available in any commercial system, it was pilot tested from 2000-2003 by a group of three schools: Stanford, Columbia and Harvard, calling themselves the InterMed Consortium(70). The consortium received a grant to practice encoding and sharing rules in GLIF, and had some success in doing so. By pilot testing the standard, the consortium learned many lessons about the scope and potential of the language, and used many of these findings to refine and improve the language. Although it is unclear if any of the shared rules were ever used in a production clinical system, the lessons learned were valuable. The types of decision support systems which can be implemented in GLIF very closely parallel the systems which Arden Syntax can implement. Like Arden Syntax, GLIF is unable to support population-based decision support systems or point-of-care information retrieval tasks.

Closely tied to GLIF, and also receiving a fair amount of attention is the Guideline Expression Language(71, 72) (GELLO) project. GLIF and Arden Syntax were both designed according to the imperative paradigm(73) of computer programming. However, object-oriented programming(74) has, over the last couple of decades, quickly outpaced imperative programming and GELLO is an object-oriented expression language for clinical knowledge representation. GELLO can replace the imperative expression languages of either Arden Syntax or GLIF, bringing them into this more modern paradigm and, to the limited extent that GELLO defines standardized terminologies, can relax some of the terminological issues affecting these systems (i.e. the curly braces problem).

Many other standards for representing decision support content in a standardized way exist. OpenClinical(75) provides a fairly comprehensive overview of guideline and knowledge modeling standards, such as Arden Syntax, GLIF and GELLO. HL7 manages a number of these projects, including Arden Syntax, as discussed above, as well as the Decision Support Service, which will be discussed in the next section, and a new and promising orderset standard.

The use of standards to represent, encode, store and share knowledge overcomes many of the disadvantages of the natively integrated decision support systems described in Phase 2 above. In particular, it provides a method for sharing the decision support content, and separates the code describing such content from the more general code which implements the clinical information system. However, standards such as the ones discussed also have some inherent limitations and disadvantages. First, there are sometimes too many standards to choose from. There are several dozen standards, in varying stages of readiness and adoption, available just to represent simple alerts and reminders. The second problem is that any encoding standard inherently constrains what a user can encode. Standards are developed for a specific purpose, and their specification is often narrowly constrained to that purpose. This works well in many cases, but can be very limiting when a user wants to go beyond the initial purpose of a standard. At the time that Arden Syntax was developed the goal was to develop a system for patient-specific event-driven decision support and, as such, this is the only type of decision support that can be encoded in Arden Syntax. This limitation is not present in standalone or natively integrated decision support systems. Since their scope is limited only by the underlying capabilities of the programming language the clinical system is implemented in and, to the extent that the chosen language is Turing-complete (as most modern languages are) any computable task can be implemented. The particular nature of this limitation has not been widely acknowledged in the decision support literature, at least in comparison to the other limitations that will be described but it is extremely significant, insofar as it limits the type of medical logic that can be modeled in certain knowledge representation formalisms. Terminology issues also significantly limited the adoption of these standards – unless clinical systems and decision support systems shared a common terminology standard, cumbersome mapping between the terminology used in the clinical system and the terminology used in the decision support system had to be undertaken. Finally, even if there were a perfect standard for sharing decision support rules and guidelines, there would remain significant unanswered questions about where such guidelines would be held, how they would be owned, who would be liable for them, how they would be evaluated, and who would keep them up to date.

1.5 SERVICE MODELS FOR DECISION SUPPORT

More recent efforts have separated the clinical information system and clinical decision support system components of an integrated decision support system and recombined them by using a standard interface. The first effort along this front was the Shareable Active Guideline Environment project (SAGE).(76, 77) SAGE placed an interface in front of the clinical system. A properly designed SAGE rule could interact with any clinical system that made this SAGE-compliant interface available. The approach that SAGE took, placing a standardized interface in front of the clinical system, has been termed a Virtual Medical Record(78) (VMR) approach. The principal advantage of this approach is that it solves the vocabulary problem – the SAGE virtual medical record exactly specifies the vocabularies that will be used to access and process the medical record, and to the extent that a clinical system uses different terminologies, it is required to provide a suitable mapping. This approach is extremely promising, however, it has some significant shortcomings. First, it makes event-driven logic more difficult. In SAGE, it is the job of the decision support module to query the clinical system for data to act on, while the clinical system is relegated to a passive role – providing data upon request. Many types of decision support are event-driven – for example, a rule that checks for correct drug dosing should only fire when a new drug is ordered, and the SAGE model makes such event logic very complex. There is no clear method for a SAGE drug dosing decision support rule to know that a new drug has been ordered, and that it should then run. Various back-channels have been proposed and prototyped which would allow a clinical system to alert a SAGE module that an event has occurred, but they have generally been too cumbersome. Also, like Arden Syntax, it requires a standard guideline format, necessarily constraining the type of decision support that can be implemented in SAGE. The SAGE project initially gathered great momentum, but it has since mostly lost steam, in part because of a belief by some vendors that it was too heavily dominated by IDX (IDX Systems Corporation, Burlington, VT, now owned by GE). Nonetheless, the approach described by SAGE is promising and it will inform in some significant ways the approach of this dissertation.

SEBASTIAN, a more recent system, has taken the opposite approach from SAGE.(79) It places a standardized interface in front of clinical decision support modules, and makes no demands on the clinical system to store data in any particular way. In this model, any clinical system which understands the SEBASTIAN protocol can make queries of centralized decision support services. SEBASTIAN began as a student project, but progress is now continuing on it through HL7, where it is being reviewed as the HL7 Patient Evaluation Service.(80) SEBASTIAN maintains most of the same advantages of something like the Arden syntax, while freeing the user from the restrictions that a statically defined knowledge representation format imposes. Moreover, since the modules are located on the Internet, they can be shared by more than one hospital, allowing for greater efficiency. However, SEBASTIAN puts a heavy onus on the consumers (i.e. clinical systems) of the services it provides. First, although

SEBASTIAN is standardized, each knowledge module is free to require any given set of data, which the clinical system must fetch and provide. Moreover, two SEBASTIAN knowledge modules may use different vocabulary standards, forcing the consumer of the service to provide the same data more than once in different encodings (and necessitating that the clinical system provide support for all the vocabulary standards chosen). Also, SEBASTIAN requires that a service consumer move patient data to the service, which some hospitals or providers may be reluctant to do. Further, because the amount of patient data needed may potentially be large, performance issues may manifest, although in early testing to this point, performance has been found to be acceptable.(79)

Both of these interface-oriented systems provide some significant advantages over the systems which require a standard representation, and both are quite promising. However, each system constrains itself to standardizing only one of the two interfaces at the junction between a clinical decision support system and a clinical system (i.e. the interface into the decision support system, and the interface into the clinical system), which limits their potential for success. Also, both systems principally look at only one clinical system and one decision support system at a time, although, in the real world, knowledge about the patient (that which is stored in a clinical system) and knowledge about medicine (that which is stored in a decision support system) can be fragmented across several, or as many as dozens of sites.

1.6 CONCLUSIONS ON THE LITERATURE

There is a clear arc in the evolution of clinical decision support. The first clinical decision support systems were standalone, while later systems were integrated. The inability to share integrated decision support systems led to attempts at standardizing knowledge representation but the limitations these attempts faced led to interface-oriented approaches. The process was evolutionary and all of these approaches had unique advantages and disadvantages, which are described in detail above. That said, there were some common limitations that almost all the approaches faced and that no single approach was able to entirely surmount: first, fixed knowledge representations systems inherently circumscribe the type of knowledge that can be represented in them; second, there are serious terminological issues; third, patient data may be spread across several sources with no single source having a complete view of the patient; and fourth, and perhaps most important, significant difficulties exist in transferring successful interventions from one site to another. Although a small number of institutions had great success with decision support, this success could not be (or was not) widely replicated in community settings. The next chapter proposes a new distributed architecture for clinical decision support which endeavors to overcome these limitations. This architecture takes advantage of burgeoning efforts to create interoperable clinical systems, and to develop a National Health Information Network (NHIN).

CHAPTER 2: TOWARDS A NEW MODEL OF CLINICAL DECISION SUPPORT

2.1 A NEW ARCHITECTURE FOR DECISION SUPPORT

This dissertation proposes a new architecture for decision support called the Service-oriented Architecture for NHIN Decision Support, or SANDS. SANDS is designed to surmount the limitations of existing architectures described in the previous section. It is hypothesized that SANDS can provide significant advantages over existing decision support architectures in the areas of transferability, scalability and integrability, defined as:

Transferability: The ability to take a known-successful intervention in operation at one site, and transfer it to another site.

Scalability: The ability to provide a wide variety of kinds of decision support (such as patient level alerts and reminders, information interventions, panel or population-wide interventions, patient-oriented decision support, etc.) within a given architecture.

Integrability: The ability to readily integrate a decision support system into a clinical system. This covers issues such as proper terminology and data-field mapping as well as issues like supportability and maintainability of knowledge content.

The purpose of this dissertation is to propose, describe, prototype and evaluate such an architecture. A loose schematic for the proposed architecture is given in Figure 6. This architecture draws on the work done for both the SAGE and SEBASTIAN projects, with significant enhancements. In this architecture, an interface is provided in front of *both* the clinical system and decision support system components, instead of in front of one or the other, as provided by SEBASTIAN and SAGE. Moreover, this architecture explicitly contemplates the case where a patient's record is spread across multiple clinical systems, and the parallel case where several clinical decision support systems are needed to fully inform a decision. The case where a patient's record is spread across several systems and needs to be reassembled to provide a complete clinical picture is the exact case that efforts to create a National Health Information Network (NHIN) are targeting and, as such, the patient data half of the architecture will draw heavily on existing developments in the NHIN space.

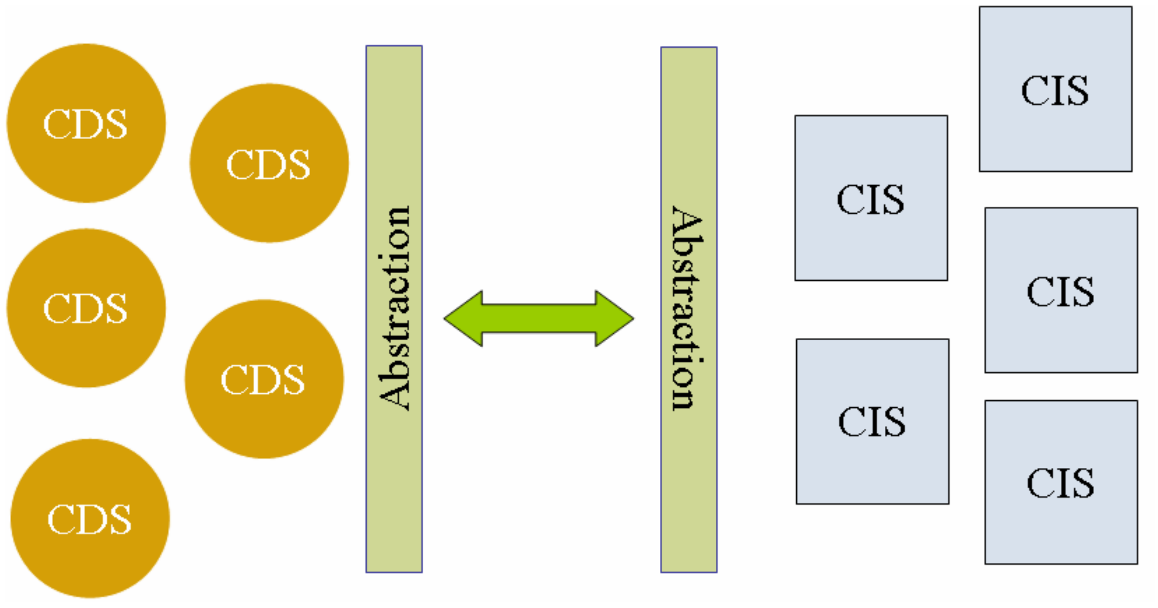


FIGURE 6 - A SCHEMATIC REPRESENTATION OF THE PROPOSED ARCHITECTURE.

On the decision support network side, one could imagine a case where a physician would like to query several different decision support service providers for different kinds of decision support for a given patient. For example, if the physician were prescribing a new drug to a diabetic patient, he or she might want to query a guideline service provided by the American Diabetes Association for the latest guidelines in diabetes management, and might also want to send the proposed prescription to a drug-interaction checking service, such as the one provided by Thomson Micromedex. This architecture also explicitly allows for the case where one decision support system queries another – for example, the American Diabetes Association might develop a decision support module for evidence-based diabetic care, and that module may in turn depend on another module, provided by the American Heart Association, that defines hypertension.

2.2 AN EXAMPLE USE CASE

To fully understand this architecture, it is perhaps best to operationalize it. Figure 7 shows a simple case of this architecture. In this case, there are two providers: Dr. Anderson, a primary care provider, who uses Epic's EHR; and Dr. Baxter, a gastroenterologist who uses the Logician EHR (now called the Centricity Physician Office EHR [GE Healthcare, Waukesha, WI]). They share a common patient in Frank Jones. Mr. Jones sees Dr. Baxter for management of severe Gastroesophageal Reflux Disease (GERD), but today presents to his primary care provider, Dr. Anderson complaining of a sore throat, which Dr. Anderson diagnoses as streptococcal pharyngitis. Dr. Anderson plans to prescribe erythromycin to treat the infection, but first asks Mr. Jones what medications he's on. He reports that he is taking Lipitor and Aspirin, as prescribed by Dr. Anderson, as well as Prevacid for his GERD, as prescribed by Dr. Baxter. Seeing no danger, Dr. Anderson initiates a prescription for erythromycin in her clinical system, but before the prescription is accepted, the decision support network is queried. A message, containing the intended prescription, as well as a pointer to Mr. Jones' record in the NHIN is sent to a drug checking service that Dr. Anderson subscribes to. This service sends a medication list query to the NHIN interface which uses its record locator service to find that Mr. Jones has records in two disparate clinical systems: those of Drs. Anderson and Baxter. The NHIN interface requests the medication lists in these systems, aggregates them and returns them to the drug checking service. This service notices, however, that Dr. Baxter's medication list indicates that Mr. Jones is actually on Propulsid for his GERD, *not* Prevacid, as Mr. Jones had indicated to Dr. Anderson. There is a very severe and potentially fatal interaction between Propulsid and Erythromycin, and the system provides this information to Dr. Anderson's clinical system which raises an alert and blocks the prescription. Although this may seem like a simple case, it's important to note that, even though the FDA engaged in a significant outreach and public relations campaign to make doctors aware of this interaction it killed at least 78 people, and is suspected in the deaths of 302 others. In the end, the FDA had to withdraw Propulsid from the market because it was unable to reliably prevent the two drugs from being co-prescribed. This decision was necessary because, in cases such as this one, where information is incomplete, this reaction is not always preventable. A decision support architecture such as the one described herein (or another safety mechanism, such as pharmacist verification) may be necessary to reduce the risk of certain drugs to a level that would justify keeping them on the market.

2.3 ARCHITECTURAL KEY POINTS

This example case highlights several key points about this architecture.

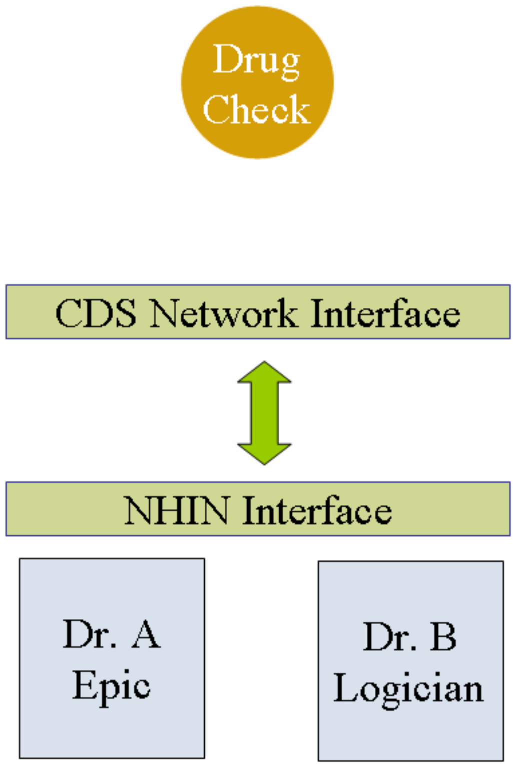


FIGURE 7 - DRUG INTERACTION CHECKING: A SIMPLE CASE OF THE DECISION SUPPORT ARCHITECTURE.

First, this architecture is defined entirely by interfaces. There are no restrictions on the internal knowledge representation approach taken by the decision support components, and there are no restrictions on the way that the clinical systems store clinical data internally. As long as the systems export the appropriate interfaces, they are compliant with the requirements of this architecture. This interface-driven approach is sometimes generically called a Service Oriented Architecture (SOA). SOAs are currently making significant inroads in the healthcare IT space. Kaiser Permanente (81), MD Anderson Cancer Center (82), the Mayo Clinic (81) and the Partners Healthcare System (83) have all announced plans to migrate their clinical systems to an SOA. None, however, have yet announced plans to fully migrate their decision support to an SOA, largely because there is no clear architecture over which to do so. This dissertation aims to fill that gap.

2.4 THE ROLE OF STANDARDS

Building an SOA for decision support requires a significant number of standards. A sample of the standards that might be required is shown in Figure 8. The standards in this figure fall roughly into two groups: healthcare informatics standards, such as HL7 (84-87), SNOMED (88), NCPDP SCRIPT (87), RxNorm (89, 90) and NDC (91, 92) which might be used to describe drugs, or transfer patient data; and SOA-related standards, such as SOAP and XML, which are used to transport data between services, and UDDI and WSDL, which are used for discovery of services, and interface definition. Depending on the application domain, other standards may also be required. Standards are not a major focus of this dissertation, in large part because robust standards harmonization activities, coordinated by the ANSI Healthcare Information Technology Standards Panel (HITSP) under contact from ONC, are currently underway (93, 94). The expected result of these activities is a set of harmonized standards ready for adoption. Because the HITSP process is ongoing, this dissertation will preferentially use standards approved by HITSP, augmented by other standards as needed. After the dissertation is complete, it is anticipated that the architecture described will be submitted for standardization, and this will be the appropriate time to consider standards related issues.

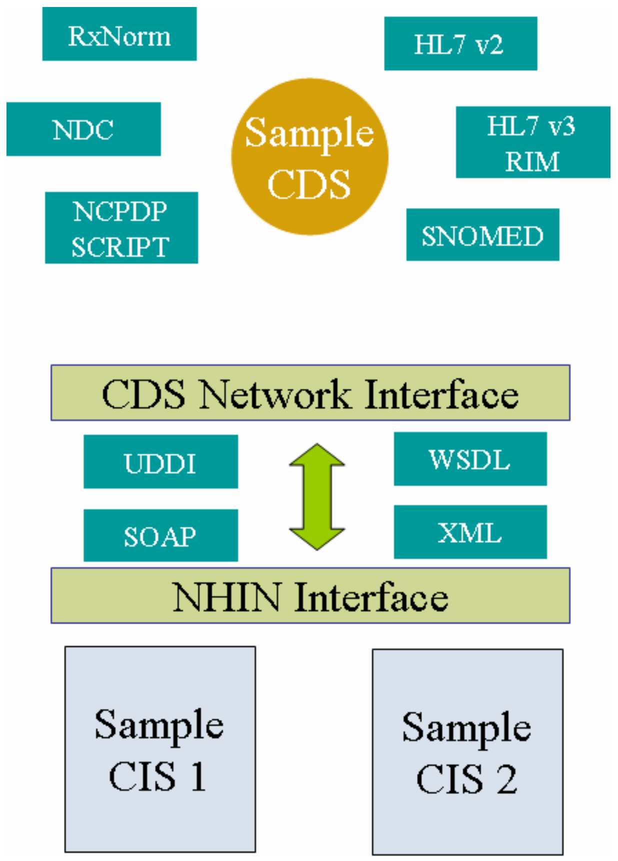


FIGURE 8 - A SAMPLE OF THE STANDARDS WHICH MIGHT BE REQUIRED FOR SUCH A DISTRIBUTED DECISION SUPPORT ARCHITECTURE.

2.5 THE NHIN INTERFACE

As discussed in the previous sections, there are two key interfaces at the core of the SANDS architecture: the NHIN Interface and the CDS Network Interface. This section gives an overview of the NHIN Interface, while the next section gives an overview of the CDS Network Interface.

At present, there is, of course, no actual NHIN, so for the sake of this dissertation it will be necessary to give some consideration to both what an NHIN might do in general terms, and also to current early efforts towards building an NHIN. Fundamentally, an NHIN is a nationwide network for sharing patient information. At a minimum, an NHIN would consist of a set of data sources, a way of querying those sources, and some system for managing privacy and security through a series of access controls. Having an NHIN enables decision support in two ways: first, an NHIN provides a more complete view of a patient's medical record than any single clinical system is likely to, and second, an NHIN is likely to present data in a standard format.

Perhaps the most comprehensive thinking about what the role and components of an NHIN might be is given in the Markle Foundation's Common Framework (95, 96). The Common Framework is a set of implementation guidance developed for both local health information exchanges (like RHIO's) as well as composite exchanges like an NHIN. A schematic of the Common Framework is given in Figure 9. This figure is reproduced from the Common Framework document (96) by license from the Markle Foundation.

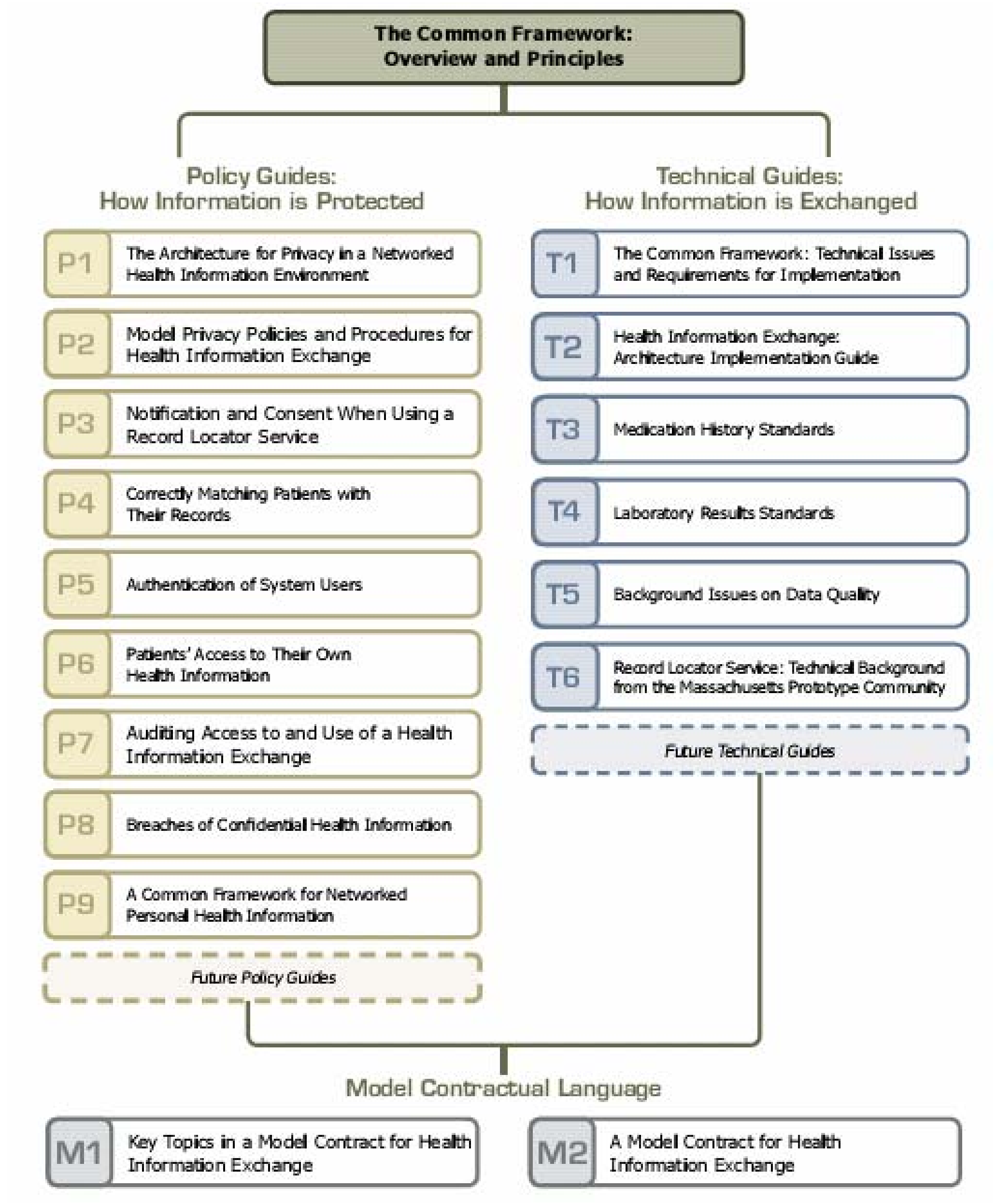

The Common Framework: Overview and Principles
Policy Guides: How Information is Protected
Technical Guides: How Information is Exchanged
P1 The Architecture for Privacy in a Networked Health Information Environment
P2 Model Privacy Policies and Procedures for Health Information Exchange
P3 Notification and Consent When Using a Record Locator Service
P4 Correctly Matching Patients with Their Records
P5 Authentication of System Users
P6 Patients' Access to Their Own Health Information
P7 Auditing Access to and Use of a Health Information Exchange
P8 Breaches of Confidential Health Information
P9 A Common Framework for Networked Personal Health Information
Future Policy Guides
T1 The Common Framework: Technical Issues and Requirements for Implementation
T2 Health Information Exchange: Architecture Implementation Guide
T3 Medication History Standards
T4 Laboratory Results Standards
T5 Background Issues on Data Quality
T6 Record Locator Service: Technical Background from the Massachusetts Prototype Community
Future Technical Guides
Model Contractual Language
M1 Key Topics in a Model Contract for Health Information Exchange
M2 A Model Contract for Health Information Exchange

FIGURE 9 - MARKLE FOUNDATION COMMON FRAMEWORK (REPRODUCED WITH PERMISSION).

As shown in the figure, the Common Framework actually presents two parallel sets of guidance: policy guides that lay out important policy questions and issues that developers of health information exchanges must consider, and technical guides that provide specific implementation guidance, specifying how Common Framework compliant exchanges should operate. The Common Framework has been adopted by a number of local information exchanges and is the basis for a prototype NHIN funded by ONC.

2.6 THE DECISION SUPPORT NETWORK INTERFACE

In addition to the NHIN infrastructure components described in the previous section, the SANDS architecture also requires a number of components, interfaces and protocols relating directly to decision support. This is the core focus of this dissertation, and most of these components are developed *de novo*. These components include:

Invocation method: Each decision support service will need to expose an API publicly, to allow clients (whether they be clinical systems, other decision support services, or users directly) to request the service. This would frequently be a triggering event in a clinical system, but could be any event that the decision support service developer makes available and a client supports.

Response Format: Many invocations of a decision support service require a response, such as an alert in response to a query to a drug interaction service. However, there is currently no formal way of describing such responses. The development of a core taxonomy of response actions and a format for describing them is a key part of this dissertation.

Discovery service: A network such as this could potentially support hundreds or thousands of distinct decision support services. Most users will rely on a human curator to sort through these services and make recommendations about the ones to use, but an automated mechanism to discover and catalog such services is also desirable. This discovery service will be based on the Universal Discovery, Description and Integration (UDDI) protocol, which is commonly used in SOAs.

Interface service: Once a desired decision support service has been located it will be necessary to determine the interface standards (API) it uses. The Web Service Definition Language (WSDL) from the World Wide Web Consortium (W3C) will be used for this purpose.

Mirroring service: Many decision support applications are real-time so performance is important. Although a pure, distributed SOA model can be optimized for performance, sometimes networking constraints (particularly latency) can limit performance. A content distribution and mirroring service, based on existing distribution systems for web and Internet content (such as the system employed by Akamai (97)) will be developed to allow for local or near-local mirroring of content with proper synchronization.

The two most critical and novel components of this architecture are the invocation method and the response format. Because of their criticality, I will first develop an empirical functional taxonomy of clinical decision support, which will serve as the theoretical underpinning for these two components. This taxonomy will be described in Chapter 3.

2.7 ADVANTAGES OF THE NEW ARCHITECTURE

There are significant advantages to using an SOA for clinical decision support. While these advantages are discussed and developed throughout the Methods, Results, Discussion and Conclusion chapters of the dissertation, it is perhaps instructive to frame the next several chapters by previewing some of them now:

1. **Modularity:** An SOA provides more modularity than other architectures, allowing for work to be distributed. With a fully realized architecture specialty societies might, for example, each produce guidelines in their area of expertise, but make them available for consumption by anyone.
2. **A Market for Decision Support:** Commercial services can play a role in an SOA, by providing services and content for which they charge a fee.
3. **Trialability / switchability:** An SOA reduces the cost and risk of trying new decision support systems: a hospital or healthcare provider could always connect to a new decision support service and try it out, but could freely disconnect if it did not perform as desired.
4. **Maximal Expressiveness:** An SOA specifies the interfaces a service must provide, but imposes no restrictions on its implementation. By contrast, standards like the Arden Syntax, GLIF and GELLO limit content developers to only those intervention types that can be expressed within the bounds of that particular knowledge representation formalism.

5. **Alignment with Interoperability:** Finally, the key advantage of this architecture is its potential to “unstick" progress on decision support, by uniting the direction of clinical decision support with promising near-term efforts to improve interoperability.

The American Medical Informatics Association (AMIA) and the American College of Medical Informatics (ACMI) recently conducted a survey to identify grand challenges and focus areas. This survey identified interoperability as the number one grand challenge for informatics and clinical decision support as the number four challenge (98). This architecture lies at the intersection of, and makes significant contributions to both these areas.

Further, ONC has taken a keen interest in how clinical decision support would work in an NHIN, and recently commissioned a national task force of decision support experts to discuss these issues and provide consensus recommendations. The author was invited to join the task force steering committee staff and is an author of its report (99). A recent meeting of ACMI also discussed these issues. Both groups focused much of their discussion on whether there should be a central repository of rules in a format such as Arden Syntax, or whether a different architecture was needed, and both groups seemed to lean, at least tentatively, towards an alternate architecture, without specifying what that alternate architecture should be. The architecture described here meets the consensus parameters.

CHAPTER 3: A TAXONOMY OF DECISION SUPPORT FUNCTIONS

3.1 A TAXONOMY OF DECISION SUPPORT FUNCTIONS

As discussed in the previous section, a rigorously developed taxonomy of decision support functions is necessary to support the development of a set of protocols and messages to be used as the decision support interface for the proposed architecture. This chapter describes such a taxonomy and how it was developed.

This chapter is based on an article titled “A Description and Functional Taxonomy of Rule-Based Decision Support Content at a Large Integrated Delivery Network”, authored by myself, Howard Goldberg, MD and Tonya Hongsermeier, MD of Partners HealthCare and Blackford Middleton, MD, MPH, MSc of Partners HealthCare and the Department of General Internal Medicine and Primary Care, Brigham & Women’s Hospital, Harvard Medical School. This paper was originally published in the Journal of the American Medical Informatics Association (J Am Med Inform Assn: 2007; 4; epub ahead of print April 25, 2007). All right, title and interest to the article remains the property of the American Medical Informatics Association ("AMIA"). This article is republished here with the express permission of AMIA.

3.2 PRIOR TAXONOMIES IN CLINICAL DECISION SUPPORT

A number of clinical decision support taxonomies have been proposed. Most are based on expert opinion or designed for use in a specific task. One widely used prescriptive taxonomy is that proposed by Osheroff, Pifer, Teich, Sittig and Jenders in their book, <u>Improving Outcomes with Clinical Decision Support: An Implementer's Guide</u> (100). They lay out a taxonomy of clinical decision support methods:

- Documentation forms / templates
- Relevant data display
- Order creation facilitators
- Time-based checking
- Protocol / pathway support
- Reference information and guidance
- Reactive alerts and reminders.

This taxonomy is very useful for considering which methods of intervention might be useful to solve a particular clinical or quality problem, however, it is not as useful for those who are seeking to develop or share a decision support system.

Other researchers have taken an empirical approach to developing taxonomies. A recently published paper by Berlin, Sorani and Sim (101) develops a taxonomy based on a review of 58 randomized controlled trials of clinical decision support systems entailing 74 clinical decision support scenarios. They identify five categories: context, knowledge and data source, decision support, information delivery and workflow. The context category describes the "setting, objectives, and other contextual factors of a system's use" and includes taxa such as *clinical setting* and *clinical task*. The knowledge and data source category looks at the *sources of clinical knowledge* (such as guidelines), and *patient data source* (EMR, direct entry into the system, etc.). The decision support category looks at the reasoning aspect of the system, such as the *type of inference* being made and the *complexity of the recommended action*. The information delivery category comprises taxa such as *delivery format* and *mode* – this category is particularly important because not all of the decision support systems reviewed in the paper were fully computerized – some, for example, resulted in printouts to be inserted into paper charts. The final category in the Berlin taxonomy is workflow, which includes taxa such as the *user of the system* (some systems target clinicians, while others are used directly by the patient), and *system-workflow integration*. This taxonomy provides an insight into the design and intent of clinical decision support systems.

Another taxonomy has been proposed by Wang, Shabot, Duncan, Polaschek and Jones, based on their experience implementing a CPOE system at Cedars-Sinai hospital (102). Their hierarchy has three levels. The top level describes the benefit of a clinical rule: *process improvement*, *policy implementation*, *error-prevention* and *decision-support*. Under these benefits are a set of domains, such as *laboratory* (under the *process improvement* benefit), *pharmacy* (under the *error-prevention* and *decision-support* benefits) and JCAHO requirements (under the *policy implementation* benefit). The lowest level of the taxonomic tree is termed "class." Classes are used to logically organize clinical rules by content type and include such elements as *drug-drug interaction checking*, *automated orders* and *guided dosing*.

The purpose of this taxonomy is to guide the organizational aspects of clinical decision support. For example, the authors used the benefit classifications in presentations about their CPOE system to help end users understand the advantages of CPOE. The domain classification is used to determine the “owner” of a piece of decision support content – for example, the pharmacy and therapeutics committee at the hospital is responsible for decision support assigned to the pharmacy domain. The lowest level, class, is useful for knowledge management and implementation tasks. Unlike the Berlin taxonomy above, which is best-suited for decision support research, this taxonomy would be most useful for applied use, in managing or administering a decision support program.

The taxonomy presented here differs from the taxonomies described above in two key ways. First, it is a functional taxonomy. It does not characterize the content or purpose of the decision support interventions in the system, as the taxonomies by Osheroff, Berlin and Wang do. Instead, it describes the functional requirements of clinical decision support – those features that must be made available to decision support systems so that they can carry out their activities. Such a taxonomy is useful for designing clinical systems, since it is generally the clinical system which exposes these features, but it is also useful for developers of decision support standards and knowledge representation formalisms, as well as for those who might wish to design a decision support service, because the elements of the taxonomy correspond to the input and output requirements of such a formalism or service. In addition to its unique perspective, this taxonomy also differs from previous work because of the breadth of its empirical basis – it is derived from a comprehensive analysis of the decision support content in use at Partners Healthcare System, a large integrated delivery network with a long history of computer-based, point-of-care decision support (103, 104).

3.3 METHODS FOR DEVELOPING A TAXONOMY

SITE AND CONTENT DESCRIPTION

Partners Healthcare System is a federation of hospitals in the Boston metropolitan area. Its original founding members were the Brigham and Women's Hospital and the Massachusetts General Hospital, both teaching affiliates of Harvard Medical School. Since its founding in 1994 Partners has grown and its current membership is described in Table 1.

Members
Brigham and Women's Hospital*†
Massachusetts General Hospital*†
Faulkner Hospital
McLean Hospital†
Newton-Wellesley Hospital
North Shore Medical Center
MGH Institute for Health Professions
Partners Community Healthcare, Inc.
Partners Continuing Care

* Founding Members
† Teaching Affiliates of Harvard Medical School

TABLE 1 - MEMBERS OF PARTNERS HEALTHCARE.

Each member hospital maintains control over its own clinical systems and uses a combination of centrally or locally managed information system resources. For example, ancillary departmental systems are managed locally, but may employ services from the centrally managed enterprise master patient index. In 2002 Partners launched an enterprise knowledge management effort with the goal of moving all decision support content in use at the sites into a centralized knowledge management portal. This portal now contains most Partners content, though a small amount has not yet been moved, and remains hard-coded in applications. This portal provides a robust environment for collaborative development and review of clinical knowledge. The portal has been described in depth elsewhere (105). The portal does not contain directly executable content. Instead it contains knowledge specifications in various forms which can be used by developers and implementers of clinical systems.

STUDY DESIGN

The authors conducted an exhaustive review of the contents of the knowledge management portal in order to develop the taxonomy. Four functional categories were identified *a priori*:

- **Triggers:** The events that cause a decision support rule to be invoked. Examples of triggers include prescribing a drug, ordering a lab test, or entering a new problem on the problem list.
- **Input data:** The data elements used by a rule to make inferences. Examples include lab results, patient demographics or the problem list.

- **Interventions:** The possible actions a decision support module can take. These include such actions as sending a message to a clinician, showing a guideline or simply logging that an event took place.
- **Offered choices:** Many decision support events require users of a clinical system to make a choice. For example, a rule that fired because a physician entered an order for a drug the patient is allergic to might allow the clinician to cancel the new order, choose a safer alternative drug, or to override the alert and keep the order as written but provide an explanation.

These categories were identified because, together, they fully describe and specify the components of an interface between a clinical decision support system and a clinical system. Similar categories are seen in other efforts, particularly knowledge representation formalisms and decision support services. For example, Arden Syntax (61) rules are broken down similarly, with the Arden Syntax "evoke" section corresponding to our trigger category, the "data" section corresponding to our input data category, and the "action" section corresponding to our interventions category.

The individual elements, or taxa, inside the categories were determined empirically over the course of the review, and each rule in the knowledge management portal was assigned to the appropriate taxa. After the review was completed the taxa were reviewed and refined using a modified card sort method. In the final model there were great similarities between the taxa identified by this methodology and those identified in other similar efforts (56, 61, 65, 106) and to concepts frequently considered in informatics, such as elements in the HL7 Version 3 Reference Information Model (84).

One challenge encountered in conducting this study was determining the unit of analysis. For the purpose of this analysis we introduce two concepts: the rule type, and the rule. It is perhaps easiest to define these terms by example. At Partners there is a single piece of logic which fires when a patient's quinidine level exceeds 7.0 μg/mL. This would be counted as a single rule type and also as a single rule. Another module, however, encodes 1,561 drug substitutions, all of which use the same logical structure. This would be counted as one rule type, but 1,561 rules. The major unit of the analysis for this paper is the rule type; however, all tables in the results section also present rule counts. The rule type was chosen because it is the unit of knowledge management: each rule type has a source, an author and one or more responsible persons, and each rule type follows a management lifecycle. This challenge of choosing a unit of analysis has been encountered in other studies (56), however, in the end it poses only a moderate challenge since the universe of taxa is the same regardless of each taxon's frequency.

3.4 DECISION SUPPORT TRIGGERS

A total of nine triggers were identified, and they are described in Table 2. The distribution of triggers is extremely skewed – the top three triggers account for 94% of all rule types (and 94% of all rules). The top trigger is *new order entered*. Many common decision support interventions, such as drug-drug interaction checking and test appropriateness verification, all fire when a new order is entered. The second most common trigger is *lab result stored*, such as a rule that alerts a clinician whenever a potassium of greater than 5 mEq/L is stored. The major difference between the top two triggers relates to their synchronicity – rules triggered by order entry are almost always synchronous – that is, their result is displayed to the clinician as part of the ordering process. Lab result triggered rules, however, are asynchronous by definition. Depending on the clinical severity of the result, and the clinical system that the rule is executing within, these rules may page a clinician, send an email, alert a nurse, generate a patient letter or simply add a low-profile flag to a patient's electronic medical record (107).

The third trigger, *outpatient encounter opened*, is fairly specialized. Almost all rules with this trigger occur in the Longitudinal Medical Record system (LMR). These LMR rules generally relate to prevention – for example, whenever a new encounter is opened, a rule fires which checks to see if that patient is up to date with current NCEP cholesterol management guidelines (85). If the patient is not, an alert is shown along with appropriate remedial actions, such as ordering a cholesterol test or starting the patient on a statin.

The remaining triggers are much less frequent, and generally self-explanatory, though a couple merit special discussion. The *user request* trigger relates to a number of guidelines and order sets which do not have any automatic trigger – instead, a user must intentionally request them. The *time* trigger has a number of different uses. For example, one rule fires every morning at 9:00 am to check the currency of lab values.

Another rule fires one week after an abnormal mammogram, prompting the clinician to contact the patient to confirm that appropriate follow-up measures are underway.

A single rule can have multiple triggers and many do. For example, one rule watches for hypokalemia in patients on digoxin. The rule is triggered whenever digoxin is ordered (a *new ordered entered* trigger), and also whenever a new potassium result is stored (a *lab result stored* trigger).

Trigger	**Rule Types**	**Rules**	**Example Rule**
Order entered	104	6777	When digoxin is ordered, check potassium.
Lab result stored	93	998	When glucose stored, check value.
Outpatient encounter opened	42	48	When a patient presents for a routine physical, order cholesterol test if needed.
User request	4	152	When user requests them, show antibiotic utilization guidelines.
Time	4	25	24 hours since admission, check for a medication list.
Admission	3	151	When a patient is admitted for congestive heart failure, offer standard therapy.
Problem entered	1	145	When asthma is diagnosed, request date of onset.
Enter allergies	1	3	When a penicillin allergy entered, check drug list.
Enter weight	1	3	When a patient's weight is entered, ensure that it is reasonable.

TABLE 2 – TRIGGERS FOR DECSION SUPPORT.

3.5 INPUT DATA ELEMENTS FOR DECISION SUPPORT

With fourteen taxa, input data is the largest category in the taxonomy. Table 3 shows the category. While still quite skewed, it has the largest spread of any category with eight of the fourteen taxa being represented by at least ten rule types. The most frequent input data elements consumed by rules are *lab results* and *medications*, exactly mirroring the two most common triggers. Many rules use both of the data elements, such as the rule described above, which monitors potassium in patients on digoxin.

The third most common data element is *hospital unit*. Many inpatient rules apply only to patients in certain units – for example, patients in the coronary care unit have more narrow cardiac parameters than patients in other units. There are also certain drugs and procedures which can only be ordered in specific areas – for example, succinylcholine can only be ordered in the ICU.

Input data element	**Rule Types**	**Rules**	**Example Rule**
Lab result / observation	126	2087	Check if latest HbA1C is >6%.
Drug list	108	4752	Active prescription for fluoxetine.
Hospital Unit	85	906	Coronary care unit.
Diagnosis / Problem	43	1587	Decrease dose of cefuroxime in patients with renal insufficiency.
Age	39	3131	Warn about nifedipine use in the elderly
Non-drug orders	15	694	Patient has an active TPN order.
Gender	12	1595	Only suggest a mammogram in females.
Family history	10	10	Suggest lipid panel more frequently for patients with family history of MI.
Allergy List	9	649	Check for a penicillin allergy when amoxicillin is prescribed.
Weight	8	1310	Suggest lipid panel more frequently for overweight patients.
Surgical history	8	8	Do not recommend mammogram with history of bilateral mastectomy.
Reason for admission	2	148	Suggest default orders when a patient is admitted for myocardial infarction.
Prior visit types	2	2	Check for ophthalmology visit in the past year for diabetic patients.
Race	1	1	Recommend a calcium channel blocker for patients with black race.

TABLE 3 - INPUT DATA ELEMENTS CONSUMED BY DECISION SUPPORT RULES.

As would be expected, a variety of demographic data are also used in decision support, such as *age*, *gender* and *race* (which is used in only a single rule, which recommends calcium channel blockers in black patients). *Family history* is also used – it is encoded at Partners, by disease and severity. The mammogram rule would recommend more frequent mammograms, beginning earlier, for a woman with an extensive family history of breast cancer. The rest of the data elements used are explained in the table. It is worth noting that there are very few natural language processing systems in use at Partners, so none of the rules used in developing this taxonomy used clinical notes or reports directly, although some systems did use specific coded findings, which are exposed as observations.

3.6 DECISION SUPPORT INTERVENTION TYPES

The interventions category is the smallest category. The most common member of the category is the most complex – *notification*. All forms of notification involve communicating a piece of information to a responsible clinical user, but these notifications can take many forms depending on the urgency of the information and the application context. These forms are described in Table 4. In addition to a variety of notification forms, notifications frequently offer the user choices. These choices represent the fourth category of the taxonomy, and are described in the next section.

	Synchronous	**Asynchronous**
Urgent	Pop-up messages	Paging, visual alerts on the hospital unit

Non-urgent	Informational messages	Email, clinical inbox

TABLE 4 – DIMENSIONS OF NOTIFICATION

After notification, *logging* is the most common intervention. Logged messages are stored by the application and are available for analysis and review but, unlike notifications, they are not shown to the user and do not have any associated choices. Logging is frequently used in surveillance rules and monitoring rules, particularly for adverse drug events and in research studies. *Providing defaults and picklists* as an intervention is frequent with drug dosing rules. The Nephros system for renal dosing (108) and the Gerios system for geriatric medication (109) use make substantial use of this response type. The remaining interventions are less frequent, and all interventions are described in Table 5.

Intervention	**Rule Types**	**Rules**	**Example Rule**
Notify	214	6748	Alert the user when a patient's potassium is >5.
Log	58	173	Log all uses of ketorolac for utilization review.
Provide defaults / picklists	21	3142	Compute recommended doses for a patient with renal impairment.
Show Guidelines	15	740	Show guidelines for use of antibiotics.
Collect freetext	8	391	Request a reason for overriding an alert.
Get approval	3	662	Send order to endocrinology when growth hormone is ordered.
Show data entry template	2	147	Request details when asthma is added as a problem

TABLE 5 – INTERVENTIONS BY DECISION SUPPORT RULES.

3.7 RESPONES OFFERED TO THE USER

As mentioned above, the offered choices category may be considered a child of the notify intervention. The members of this category are listed in Table 6. The most common offered choice is to *write an order*. This comes up in a variety of clinical workflows – for example, Partners has a therapeutic substitution rule which recommends famotidine when other equitherapeutic but more expensive H_2-receptor antagonists are ordered – this recommendation of famotidine would qualify as an order suggestion. This type also occurs in lab result oriented workflows. For example, when a low potassium value is stored for a patient on digoxin, clinicians are given the option of ordering potassium supplementation. The next two most frequent choice types, *defer warning* and *override rule/keep order* both dismiss the notification received without changing the user's current course of action – the defer warning choice dismisses the warning for a period of time, while the override rule choice dismisses it more permanently – until the condition worsens, or until the action which caused the notification is repeated. The next four choices: *cancel existing order*, *cancel current order*, *edit current order* and *edit existing* order all primarily occur in drug-drug interaction rules. When a clinician enters a potentially interacting order, he or she is offered the choice of canceling or editing either the new drug order, or the existing order.

The *set allergies* choice most frequently occurs in response to interventions which suggest a drug. When the system suggests, for example, a statin, the user is offered the choice to turn the suggestion down because the patient is allergic to that statin. In addition to dismissing the alert, this also adds the appropriate allergy to the patient's allergy list. The next two choices: *write letter* and *write note* provide starting text for results letters to patients and for progress notes. They are primarily used in the LMR SmartForm module at Partners (110). Finally, the *edit problem list* choice is used to add or remove items from the problem list, while the *enter weight, height and age* choice is used to query the user for this information, generally before proceeding to a calculation or inference which requires this information.

Offered Choice	Rule Types	Rules	Example Rule
Write order	63	2059	Change a ranitidine order to famotidine.
Defer warning	47	94	Allow the user to defer a warning for 24 hours.
Override rule / keep order	47	3014	Keep an order which triggered a low-severity drug interaction rule.
Cancel existing order	30	240	Discontinue an existing order for fluoxetine when it is flagged as duplicating a new order paclitaxel.
Cancel current order	29	3110	Cancel an order for furosemide in a patient with a sulfa drug allergy.
Edit current order	26	1538	Change the dose of an order for 16g of acetaminophen.
Edit existing order	23	42	Reduce dixogin when patient is hyonatremic.
Set allergies	14	20	Decline a suggestion to order atenolol because the patient is allergic.
Write letter	7	86	Send a letter to a patient with a normal mammogram.
Write note	4	23	Provide default text for a note on a patient with an elevated LDL.
Edit problem list	4	4	Remove hypertension from the problem list, in response to a suggestion for antihypertensive therapy.
Enter weight, height or age	3	787	Allow user to enter weight when ordering a drug with weight-based dosing.

TABLE 6 - OFFERED CHOICES AS PART OF NOTIFICATION INTERVENTIONS.

3.8 COMPARISON TO COMMERCIAL SYSTEMS

In order to understand the generalizability of this taxonomy, we presented it to several commercial clinical system vendors and asked them to compare it to the features available in their own clinical systems. Overall, the taxonomy appeared to map well to the functionality available in most clinical systems, and was frequently a superset of the functionality available. This process led to two clarifications of the taxonomy:

- The *observation* input data element can contain any structured data element – not just a lab result. For example, nursing documentation entered into a flowsheet and patient data entered into a structured template by a physician would both qualify as observations.
- The exact timing of triggering events is left to the discretion of the clinical system vendor. For example, ordering a new drug would be a triggering event in the taxonomy, but whether this event should fire when the drug is selected from a picklist, after the dose and instructions are entered or at the moment the order is signed is left to the discretion of the clinical system implementer.

We also compared the taxonomy to an emerging effort from the Leapfrog Group to certify the decision support capabilities of CPOE systems (114). Given the two clarifications above, the taxonomy would be sufficient to describe the clinical decision support expected in the certification process.

Finally, we compared the taxonomy to the HL7 Version 3 Reference Information Model (RIM). The RIM is an attempt at building a unified information model for clinical data, developed over the past decade. The RIM is controversial (115, 116), and has not seen wide adoption, but it is probably the most complete model for representing medical data available. We employed a two-way mapping process – first, we mapped the taxonomy elements to the RIM, and found that all of them could be adequately represented using the data model. Second, we attempted to map back from the RIM to the taxonomy. We found that certain components of the RIM could not be fully represented by the taxonomy. For example, the RIM Act hierarchy is much larger and more comprehensive than the trigger category of the taxonomy, and any RIM Act could conceivably be a trigger for clinical decision support. How, exactly, to reconcile this is a difficult question. It would certainly be possible to extend the taxonomy to include any particular act of interest, but it's not clear that every act is relevant for clinical decision support. One approach, taken by the HL7 Decision Support Service project and, to a lesser extent, with GELLO is simply to declare the space of events and elements usable for decision support to be the complete RIM. However, no current EHR supports the RIM for its data model and it is unlikely that any EHR will do so soon, so we find value in constraining the taxonomy to include those elements which we have seen used for clinical decision support, or which are supported by current systems, while recognizing that, as decision support evolves and matures, it may be necessary to extend the taxonomy to include new approaches.

3.9 CONCLUSIONS AND IMPLICATIONS

This analysis indicates that a very large amount of decision support can be accomplished with a fairly small number of functional constructs across a finite set of categories. This suggests that the problem of integrating decision support into clinical systems, though non-trivial, should be tractable, since these functional dimensions can be loosely translated into functional requirements and specifications for clinical applications and knowledge representation formalisms.

The taxonomy described here has a variety of applications. One major and immediate use is the development of knowledge representation standards. This is a rich field, and a variety of formalisms, such as Arden Syntax (61), GLIF (66) and GELLO (72) are available. It would be a productive exercise to see whether all of the taxa identified here could be properly mapped by each of these formalisms, and the taxa could likewise serve as a roadmap for future development of knowledge representation standards. For example, as mentioned earlier, Arden Syntax can map many elements of our taxonomy, but there are gaps – while we identify seven interventions (and twelve associated choices), Arden Syntax only covers one: *notify* (which it terms "write").

Beyond knowledge representation standards developers, this taxonomy may be of interest to the broader standards community. It seems sensible that each trigger and data element identified should be representable according to a suitable message and vocabulary standard but this is not currently possible. As an example, while good vocabulary standards are currently available and in use for drugs and lab tests, there is much less standardization for data elements such as family history, allergies and problems. Given the data presented here, this is, perhaps, reasonable since drug and lab data are more frequently used than the other data elements. But in a perfect world all data elements could be represented according to a standard and the results described here may help in prioritizing the development and adoption of such standards.

This taxonomy should also be useful to developers and implementers of clinical information systems and to clinical knowledge providers. Closely related to this use case is certification. A clinical system that implements features satisfying all of the functional dimensions described in this paper is likely to be capable of a fairly comprehensive range of decision support, so these dimensions may be a useful starting point in framing certification requirements for decision support, just as the HL7 EHR definition (111, 112) was used as a starting point for the CCHIT ambulatory system certification criteria.

We also hope that the taxonomy described here is useful for decision support researchers. It is one tool for evaluating the generalizability of new decision support systems and knowledge representation formalisms. Some of the most exciting developments in decision support are likely to occur at the edge of the region of functionality defined by this taxonomy, necessitating revision and expansion as new areas are explored and new foci develop.

LIMITATIONS

The major limitation of this study is that it looks at content in only a single integrated delivery network. Although this federation has a variety of hospitals, ranging from major academic medical centers to small community hospitals, there is some degree of homogeneity across the institutions. We are aware of certain classes of decision support in use at other institutions which cannot be fully described using this taxonomy. For example, the University of Utah has a decision support system for ventilator control (113). This system can automatically tune ventilator parameters: a function not included in the intervention or offered choice aspect of the taxonomy described in this paper, so this taxonomy could not fully map such systems. This is an inherent limitation of any empirically developed taxonomy – it can only include those taxa found in the site or sites on which it is developed.

FUTURE DIRECTIONS

In future work, we intend to extend this taxonomy to other institutions with two primary goals. The first goal of this extension is to measure the generalizability of the taxonomy, and to measure the extent to which it can successfully describe content in other settings. The second goal is to extend the taxonomy to include new taxa in use at other institutions, and also to generalize it beyond rule-based knowledge to include other forms of decision support, as well as other elements, such as the targeted actor or relevant clinical situations and workflows.

We also intend to begin mapping the categories and taxa identified here to currently available message and vocabulary standards as a way to assess the adequacy of the current standards base for use in decision support systems. A complete analysis of the standards landscape through the lens of decision support would be useful for standards developers.

CHAPTER 4: DEVELOPING A PROTOTYPE (METHODS)

4.1 INTRODUCTION TO THE PROTOTYPE

Given the goal of developing a working architecture for clinical decision support across an NHIN, one faces the question of how, exactly, to develop such an architecture. Some architectures and standards have been developed in a vacuum (that is, without a real, working prototype). Experience suggests that development of any architecture is most likely to be successful when it proceeds in parallel with development of a prototype. There are bound to be challenges, edge cases or special requirements which simply cannot be anticipated when an architecture is developed by simply writing it down.

To that end, as I proceeded through research on the SANDS architecture, I simultaneously developed a working prototype of the architecture, as well as a prototype EMR. Together, these prototypes provided useful tools to test assumptions about the architecture and they also helped to further specify the architecture. While the architecture is fully specified in Appendices A and B, no written specification can be fully specified without ambiguities. These prototypes, which will be made available publicly under an open source license, help increase the specificity of the architecture's description. This illustrates a key point about this project's structure: while the core contributions of this project are theoretical (the development of a taxonomy and architecture), these theoretical contributions are complemented by corresponding applied contributions (the prototypes).

4.2 OVERVIEW OF THE NHIN

As described in Chapter 2, the architecture has two interface facets: the patient data interface (here, the NHIN) and the decision support interface. The decision support interface is described in the next sections. Because there is currently no actual NHIN, I chose to use a prototype NHIN. The Office of the National Coordinator for Health Information Technology funded four consortia (led by Accenture, IBM, Northrop Grumman and CSC) to develop prototype NHINs. Each prototype was required to connect local exchange efforts in three distinct geographic markets. For the SANDS prototype I interfaced with the prototype developed by CSC and the Markle Foundation which unites exchanges in Indianapolis, Massachusetts and Mendocino, CA. This prototype was selected because it was the most mature at the time, had the greatest diversity of local exchange architectures and because it was freely available.

4.3 PATIENT DATA CLASS LIBRARY

RATIONALE FOR A PATIENT DATA CLASS LIBRARY

As discussed in the chapter on taxonomies, most decision support interventions will require two types of data to make their inferences. First, inferences generally require data which describes the context in which inferences are being made. Second, inferences also frequently require more general background data about the patient, beyond the current context. For example, a drug interaction checker needs to know the drugs that are about to be ordered (the contextual data), as well as the drugs a patient is currently taking (the background data). Without both types of information no inference can be made.

In the SANDS architecture contextual data comes from the invocation but background data comes from the NHIN. The advantages of using the NHIN for background data have already been described, but the key advantages are the ability to consolidate patient data from multiple sources and the standardized view of patient data that an NHIN affords. While, fundamentally, the patient data interface is the NHIN, as I developed prototypes it quickly became clear that parsing raw data from the NHIN was time consuming. Thus, I developed a patient data class library, described in this section, which takes data from the NHIN and processes it into a form more usable for common decision support tasks. This class library is in the spirit of a virtual medical record or VMR (78). While the class library, itself, is read-only, since it represents a view of data in the NHIN, new contents can be added simply by publishing data to the NHIN, in accordance with the mechanisms that the NHIN itself provides.

The foundation of this class library comes from the *data elements* section of the taxonomy presented in Chapter 3. The Patient Data Class Library makes each of these elements available to developers of decision support systems. In addition to greater ease-of-use the Patient Data Class Library also allows for more expansive prototyping of decision support intervention. Because current NHIN prototypes are in early stages they do not always support the complete complement of data types described in the taxonomy. However, the Patient Data Class Library provides a way around this limitation. In addition to its built-in support for populating data elements from the NHIN, it can also be populated with coded test data for use in experiments. This makes it possible to prototype decision support systems that require data not yet available in early stage NHIN prototypes. This feature also allows us to create custom test patients for certain decision support interventions. For example, the NHIN prototype did not contain any pediatric patients, but one test case we developed was for diagnostic decision support with a special focus on pediatric patients. We were able to use the test-data feature of the Patient Data Class Library to test such interventions on simulated pediatric patients.

ELEMENTS OF THE PATIENT DATA CLASS LIBRARY

The specific elements of the Patient Data Class Library are:

Demographics and Vital signs:

- First Name
- Last name
- Gender
- Race
- Date of birth
- Weight
- Height
- Care setting

Clinical information:

- Observations
- Drugs
- Problems
- Family history
- Procedure history
- Allergies

- Other orders

For each of these elements one or more standard representation forms were chosen. Where possible these representation forms were based on standards approved by the Health Information Technology Standards Panel (HITSP) or the Consolidated Health Informatics working group (CHI) of the Federal Health Architecture (FHA) project, a coalition of federal agencies which work together to choose standards for sharing amongst federal agencies (117). The efforts of CHI have recently been mostly supplanted by HITSP, but there are cases where CHI standards are available, but corresponding HITSP standards have not yet been chosen.

Storing the patient's first name and last name is very simple because these are simply free-text string elements. Gender and race are both stored according to enumerated types defined by HL7, the "HL7 Gender Vocabulary Domain" and the "HL7 Race Vocabulary Domain". These vocabulary domains are intentionally extremely inclusive and are designed to be able to represent any patient's race or gender, regardless of whatever special conditions may apply.

Date of birth is stored as an HL7 formatted date (such as 20070405), while weight and height are made available in both metric and standard units. Weight and height are special data elements as patients often have serial weight and height measurements stored. In such cases the most recent weight and height are available as discrete elements, but all historic weight and height measurements are made available in the observations section of the class library. The observation section also allows metadata to be encoded; for example, whether the height and weight are patient-reported or whether they were measured in the doctor's office. Clinical decision support interventions that make significant use of these data elements should process the observations section directly to determine the best weight and height to use in their inferences or calculations.

The final element of the demographics and vital signs section of the Patient Data Class Library is care setting. This describes the current care setting for a patient; for example, whether the patient is in the doctor's office for an ambulatory visit, is in a skilled nursing facility, or is currently admitted to the intensive care unit. In the case where more than one care setting applies to a patient at a given time, the highest acuity care setting is used. These care settings are encoded according to the HL7 "Dedicated Clinical Location Role Type."

STORING OBSERVATIONS

Observations, which include such data elements as lab results, vital signs, nursing documentation and structured data entered by a physician are all made available through the Patient Data Class Library. For the current implementation of the library these elements are available in a form derived from the EHR-Lab Interoperability and Connectivity Specification (ELINCS) (118).

ELINCS was recently the subject of some controversy during a meeting of the American Health Information Community. Most current lab systems and EHR's store and communicate observations in the ELINCS format. However, ELINCS is based on version 2.4 of the HL7 specification, and more recent versions of HL7 (versions 2.5 and 3.0) are already available. The HITSP recommended adoption of HL7 version 2.5 for lab results. This resulted in a protest by the American Clinical Laboratory Association (ACLA) (119), which requested that ELINCS be chosen instead. The controversy is nearly resolved, as HL7 has taken over the development of ELINCS, and bringing it up to HL7 2.5. A final harmonized implementation guide is expected by May, 2007. In the meantime, CCHIT has announced that they will accept either ELINCS or HL7 version 2.5 compliance as a part of their certification process. Because ELINCS is more widely used at present, and because the current NHIN prototypes use it as their format for exchanging observations, ELINCS was chosen as the basis for observation representation in the Patient Data Class Library. As future NHIN prototypes migrate toward a newer version of HL7 standard it will be a simple matter to convert the translation mechanism from reading ELINCS formatted observations to reading observations stored in the newer, harmonized format.

Each observation made available through the Patient Data Class Library contains a timestamp, a Logical Observation Identifiers Names and Code (LOINC), a value and, where applicable, the units that value was measured in, a normal range and a result flag indicated whether or not the value is abnormal. LOINC is a widely used vocabulary standard for observations which was developed at the Regenstrief Institute (120, 121), and includes codes for laboratory results, such as "chemistry, hematology, serology, microbiology (including parasitology and virology), and toxicology ", as well as clinical observations, including "vital signs, hemodynamics, intake/output, EKG, obstetric ultrasound, cardiac echo, urologic imaging, gastroendoscopic procedures, pulmonary ventilator management, selected survey instruments, and other clinical observations" (122).

The LOINC vocabulary standard is key to the interpretation and utility of observations in the Patient Data Class Library, and ensures that observations provide not only syntactic but semantic interpretability.

STORING DRUG INFORMATION

Drug information in the Patient Data Class Library is stored according to national drug code (NDC), name of drug, dosage instructions, start date, end date and the date that the prescription was last filled. Selecting an appropriate drug vocabulary system was especially difficult and is discussed more fully in section 7.2. However, for most clinical decision support inventions, the current mode of storing drug data is sufficient; and, where more precise data is required, a decision support invention can bypass the Patient Data Class Library and query the NHIN directly.

STORING PROBLEMS

Patient problem list data includes a code describing the problem, a start date, end date, status, verified date, and comments. Problems can be encoded either according to the ICD-9 or SNOMED vocabularies with SNOMED strongly preferred. ICD-9 is available as a choice only because many clinical information systems provide problem data in ICD-9 format. It is important that any clinical decision support intervention be able to interpret problem list entries in both ICD-9 and SNOMED, or that such an intervention takes advantage of a translation layer to convert between the two problem vocabularies. Several such translation systems are available, including a commercial mapping developed by the American Health Information Management Association (AHIMA) and the UMLS Metathesaurus, produced by the National Library of Medicine (123).

FAMILY HISTORY

Family history items are actually special cases of the problem element type. Each family history element is composed of a problem (in this case, a problem that a family member rather than the patient had), the relationship of the person suffering from the problem to the patient, the vital status of the problem sufferer (alive, dead or unknown), his or her current age, and age at diagnosis. Relationships are encoded according to the HL7 "Personal Relationship Role Type."

PROCEDURE HISTORY

Procedure history elements are composed of a code (either a Current Procedural Terminology (CPT) code, or a SNOMED code), indication, service date and comment. In most current prototypes, CPT codes are used to describe the procedure performed and at least anecdotally CPT codes appear to be sufficiently expressive. The indication is stored as a problem type as described above.

ALLERGIES

Perhaps the most difficult data elements to represent in this Patient Data Class Library is the allergy. Current standards for exchanging data on patient allergies are immature. The most detailed recommendation about allergy representation comes from a recommendation made by CHI to the Secretary of Health and Human Services. This recommendation has been reviewed by the National Committee on Vital and Health Statistics, although adoption of the standard is extremely limited. The recommendation portion of the document actually recommends 23 separate vocabulary standards for encoding the three key elements of an allergy description: the allergen, reaction and severity. The Patient Data Class Library uses a simplified system for encoding allergies, employing SNOMED to describe the allergen, reaction and severity. There are currently efforts underway, through HITSP, to harmonize allergy standards and release a single, definitive implementation guide for allergy encoding. As these standards mature, the Patient Data Class Library will be extended to match developments in the allergy domain.

OTHER ORDERS

The final area of the Patient Data Class Library is other orders. This includes all non-drug orders, including laboratory orders, nursing orders and procedure orders. Laboratory orders are encoded according to ELINCS, nursing orders according to HL7 version 3 acts, and procedure orders according to CPT. The choice of acts for nursing orders was difficult, as there is no widely used vocabulary standard for nursing orders. However, there is currently a promising effort underway to integrate a nursing order terminology system developed by Susan Matney at Intermountain Healthcare (124) into the HL7 version 3 RIM Act hierarchy.

CACHING

Early tests of the patient data class library revealed unacceptable performance, primarily due to very slow fetch time of data from the NHIN prototype (fetching all data for a patient took 6-10 seconds). Analysis revealed that the delays were not primarily due to network latency, data transfer time or XML parsing overhead, but, instead, were a function of the current NHIN prototype's reliance on relational databases to store patient data. Many clinical systems use variants of MUMPS (125), an efficient hierarchical database which stores all the data for a single patient contiguously, allowing for quick access. However, the NHIN prototype uses relational databases, which store data in tables by type. So, for example, all the lab results for all patients would be stored in a single table, and all medications would be stored in another table. When it comes time to assemble all the data for a given patient, these tables must be searched, and records pertaining to that patient must be located and extracted. This process can be sped up significantly with intelligent indexing strategies, but relational databases have a hard cost of multiple hard disk seeks whenever patient data is stored non-contiguously.

To work around this problem, I implemented a caching strategy. The first time data is requested from the Patient Data Class Library, the library fetches that data from the NHIN and caches it. The library actually caches two different data elements: first, the entire raw response from the NHIN, and second, a materialized instance of the Patient Data Class Library. This allows client applications to avoid re-querying the NHIN, regardless of whether they access patient data through the class library, or by directly parsing the NHIN response.

Because the SANDS architecture is distributed, it would be inefficient if every node had to keep a cached copy of the patient's data. Instead, a wide-area distributed hash table is employed. The hashing function is implemented by the patient data class library using the NHIN query parameters, and the table is implemented by the memcached system (126). The memcached system is widely used for data-driven web and web-service applications, such as Wikipedia. One important question with any caching strategy is currency, and two approaches are used in the prototype to help ensure that data retrieved from the cache is current. First, all data entered into the cache is given a short expiration time (a few hours), and second, when new data is stored for a patient, the cached data objects for that patient are explicitly expired from the cache. In the present implementation, the clinical system is responsible for forcing expiration, but in the final form this would likely be a function of the record locator service. When the RLS received notice of new patient data from a clinical system, it would automatically expire that patient's cached data objects (or possibly even force an update of them).

4.4 DECISION SUPPORT SERVICE INTERFACE

In addition to the Patient Data Class Library the architecture defines a decision support service interface. This interface is fully described and specified in Appendices A an B, but some of the critical design decisions are described in this section. The decision support service interface consists of two components: one for invoking decision support services, and the other for returning structured interventions. Both components derive directly from the taxonomy presented in Chapter 3 in order to maximize their applicability to real clinical decision support scenarios.

DECISION SUPPORT SERVICE INVOCATION INTERFACE

The decision support and service invocation interface is derived from the trigger axis of the decision support taxonomy described in Chapter 3. This interface defines the way in which decision support services are invoked according to triggering clinical events. The general format for invocation is:

```
RULE(RULESET, PATIENT_ID, ATTRIBUTES)
```

Consider, for example, a decision support service designed to alert her clinician when an extremely high or low lab value (sometimes called a panic value) is stored. This service will be called whenever a new lab result is stored by a clinical system, so the lab result stored trigger (as described in Appendices A an B) would be used:

```
LABRESULTSTORED("123456","RESULT=2697-1");
```

The two arguments passed to the Lab Result Stored invocation are an identifier for the patient (keyed to the NHIN Master Patient Index by ID 123456), and a LOINC code describing the result is also passed to the service. In this case, the code is 2697-1, which is the code for a serum potassium value. It is important to note that the actual result is not passed along with the invocation. It is the responsibility of the decision support service to fetch whatever patient data is needed through the standard NHIN interface, thus freeing the calling clinical system from having to predict what data elements the clinical decision support service needs to make an inference.

The invocation interface is built on top of standard Web service protocols, particularly SOAP, WSDL and UDDI. SOAP is a format for making remote object-oriented function calls. The invocation calls shown above look like local function calls, but they actually rely on SOAP to transport the function call to a remote service, and also to transport the result of that remote function call back to the caller. SOAP toolkits are available for most modern languages, including C, C++, C#, Java, PERL, Python, Visual Basic, PHP and Ruby. These toolkits make the details of calling remote functions transparent to the users. SOAP is, in turn, dependent on the WSDL standard, a way of describing remote functions. So, for example, the fact that a remote method called LabResultStored is provided by a decision support service, and that that method requires two arguments is encoded in the WSDL format, which SOAP uses to encode and decode function calls. The UDDI protocol provides discovery and description services, so that new web services can be found by potential clients. All three of these standards are, in turn, built on top of HTTP, an application layer protocol which itself is built on the well-known TCP and IP protocols.

While the service definition specifies a variety of common invocation methods, it is important to note that, ultimately, the form of invocation and arguments required is determined by the provider of a decision support service. This provides significant flexibility as new kinds of decision support workflows can be developed within the framework of the SANDS architecture. That said, where possible, it is certainly preferable to use the standard invocation patterns as these are most likely to be supported by hooks in clinical systems.

STRUCTURED INTEVENTION INTERFACE

The expected result of a call to any decision support service is either instructions to take no action or a set of one or more interventions describing actions to be taken automatically by the clinical system or proposed to the user. The interventions and the choices offered axes of the decision support taxonomy described in Chapter 3 form the basis of a structured message format for describing decision support interventions. It is perhaps easiest to understand a structured intervention interface by looking at an example:

```
<decisionSupportResponse engine="TestEngine"
                         ruleset="OutpatientVisit"
                         time="2007-01-31T12:12:05-05:00">
    <notify severity="3"
            text="Patient has diabetes and most recent LDL (144) above target (100) and
                  not on an HMG CoA-reductase inhibitor.  Recommend HMG CoA-reductase
                  inhibitor if not allergic."
            guid="35b6c020-3e95-11db-a98b-0800200c9a66">
       <writeOrder choiceID="1">
          <orderable>simvastatin</orderable>
          <dose>20</dose>
          <doseUnit>mg</doseUnit>
          <frequency>qd</frequency>
          <number>90</number>
       </writeOrder>
       <addAllergy choiceID="2" allergenCodeSystem="mesh">
           D27.505.519.186.071.202.370
       </addAllergy>
       <addProblem choiceID="3">266468003</addProblem>
```

```
      <enterReason choiceID="4">Decline suggestion: Other Comorbidities</enterReason>
      <enterReason choiceID="5">Decline suggestion: Cost of Treatment</enterReason>
      <enterReason choiceID="6">Decline suggestion: Patient refuses</enterReason>
      <removeProblem choiceID="7">73211009</removeProblem>
      <deferNotification choiceID="8">
            <deferTime>P5D</deferTime >
            <deferTime>P1M</deferTime >
            <deferTime>P3M</deferTime >
      </deferNotification>
   </notify>
</decisionSupportResponse>
```

This example describes a menu of interventions that might be taken for a diabetic patient with hyperlipidemia who is not currently on a statin (a class of drugs used to lower cholesterol). This response consists of a single intervention called “notify”. The “notify” intervention has a severity (in this case 3, meaning low severity), a text component to be displayed to the user (in this case “Patient has diabetes and most recent LDL (144) above target (100) and not on an HMG CoA-reductase inhibitor. Recommend HMG CoA-reductase inhibitor if not allergic.”), and a number of choices that would be offered to the user, including the ability to order a statin, add an allergy to statins if the patient is allergic, remove diabetes from the patient’s problem list, defer the warning, or decline the suggestion for a variety of other reasons.

While the structured intervention message format describes the overall behavior that a clinical system should carry out, it is important to note that it does not describe the precise way that the clinical system carries that behavior out. For example, consider an alert triggered by a patient’s rising potassium value. The decision support service would send a Notify event, but the clinical system would have to determine what to do with it. If the responsible clinician is currently logged in to the computer the clinical system might provide a pop-up or other contextual alert that a notification is available and allow the clinician to choose a response from the provided menu.

However, if the responsible clinician is not logged in, the clinical system might instead page him or her with the information.

One notable element of the intervention is the GUID, or globally unique identifier. A GUID is a guaranteed-unique serial number. Decision support services have the option of passing a GUID along with their response to a query. If they do, the calling clinical system is responsible for returning the choice taken by the user along with the GUID to the decision support service that provided the intervention. This information can either be used for further decision support or for statistical purposes, such as evaluating the usefulness of the decision support intervention. For example, if the designer of the intervention described here discovered that patients were often refusing the intervention because of cost reasons, he or she might revise the intervention to use a less expensive drug.

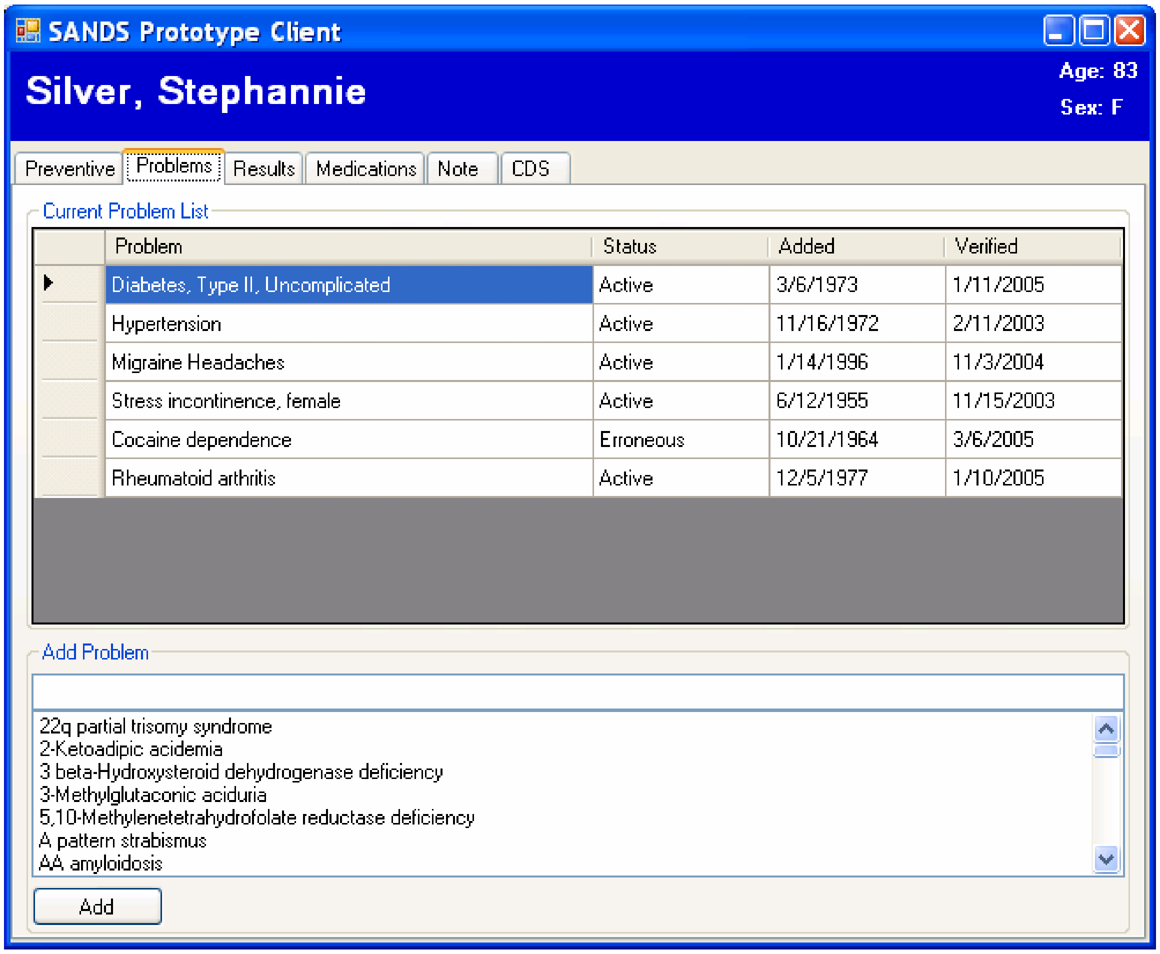
4.5 A REFERENCE PARSER

The formal service definition in the appendix is sufficient to fully describe the representation format used by that standard, both with regards to triggers and to structured intervention responses. However, experience suggests that interpretation of such a standard will be most faithful when a working implementation of the standard is also provided to interested developers. As such, a reference parser developed in C# will be made available under an open-source license. Anyone interested in developing a service to participate in this architecture should ensure that the results that their service provides can be successfully interpreted by this reference parser. Anyone interested in developing a clinical system that parses results of decision support services should use the reference parser as the foundation or at least ensure that the parser that they develop has the same behavior as the reference parser.

The current reference parser is intentionally designed to be very simple in operation. It takes an XML document, insures that it is well formed and valid, and then attempts to render the decision support response in a textual format. It contains checks for a variety of common errors that might be made in the output format. Because most decision support services provide only a small number of possible responses it should be practical to verify those responses against their reference parser and ensure that the desired behavior is observed.

4.6 PROTOTYPE ELECTRONIC HEALTH RECORD

To showcase this decision support architecture and service interface, I developed a prototype electronic health record, called the SANDS Prototype Client. The SANDS Prototype Client has most of the functionality of a regular electronic health record, along with two special features. First, the SANDS prototype is unique amongst electronic health records, because instead of having its own internal data store, it accesses the NHIN prototype to retrieve patient information. It features a problem list viewer and entry system, shown in Figure 10, a results viewer shown in Figure 11, a medication list viewer and entry system shown in Figure 12 and a progress note editor shown in Figure 13. All of this data is read from the Patient Data Class Library, which, in turn, reads data from the NHIN prototype.



SANDS Prototype Client

Silver, Stephannie

Age: 83
Sex: F

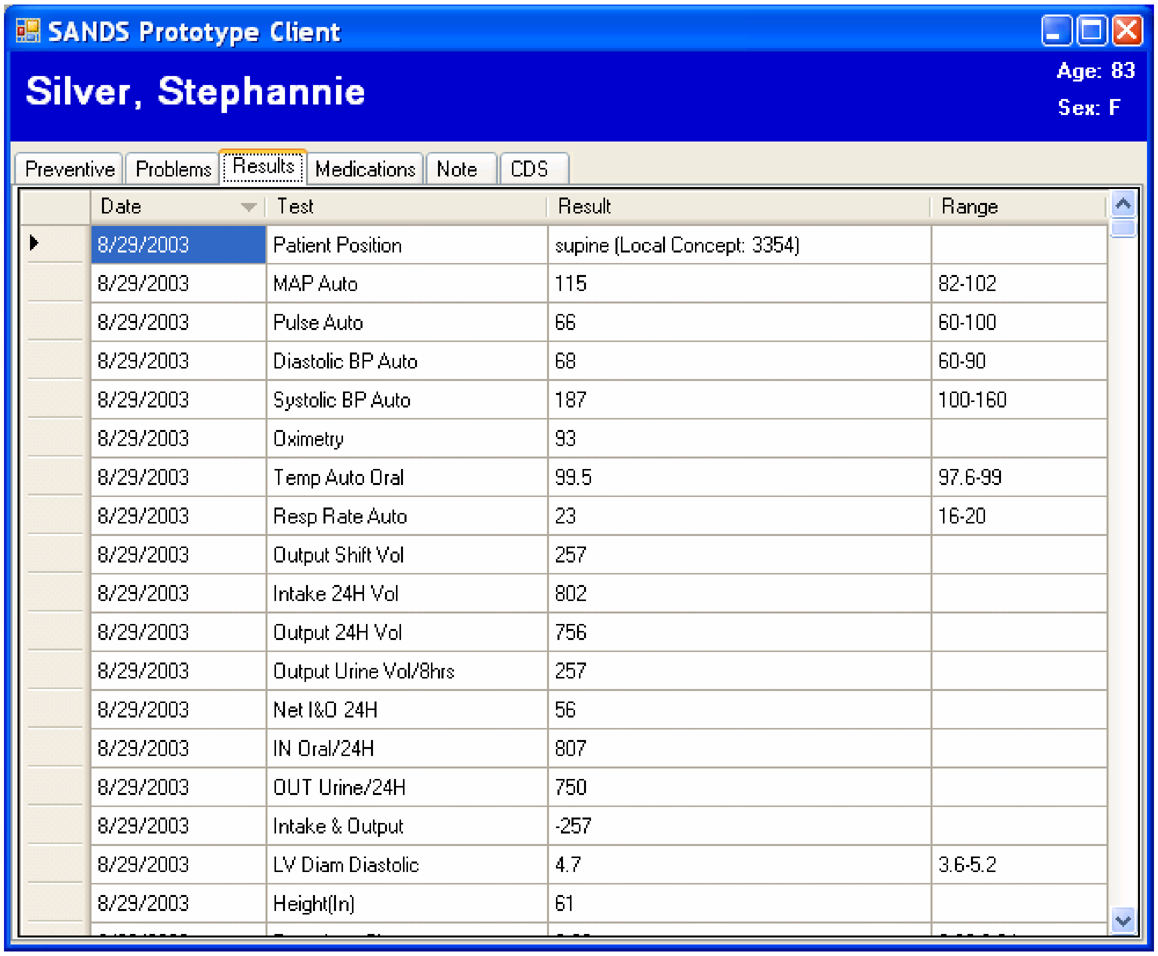
Current Problem List

Problem	Status	Added	Verified
Diabetes, Type II, Uncomplicated	Active	3/6/1973	1/11/2005
Hypertension	Active	11/16/1972	2/11/2003
Migraine Headaches	Active	1/14/1996	11/3/2004
Stress incontinence, female	Active	6/12/1955	11/15/2003
Cocaine dependence	Erroneous	10/21/1964	3/6/2005
Rheumatoid arthritis	Active	12/5/1977	1/10/2005

Add Problem

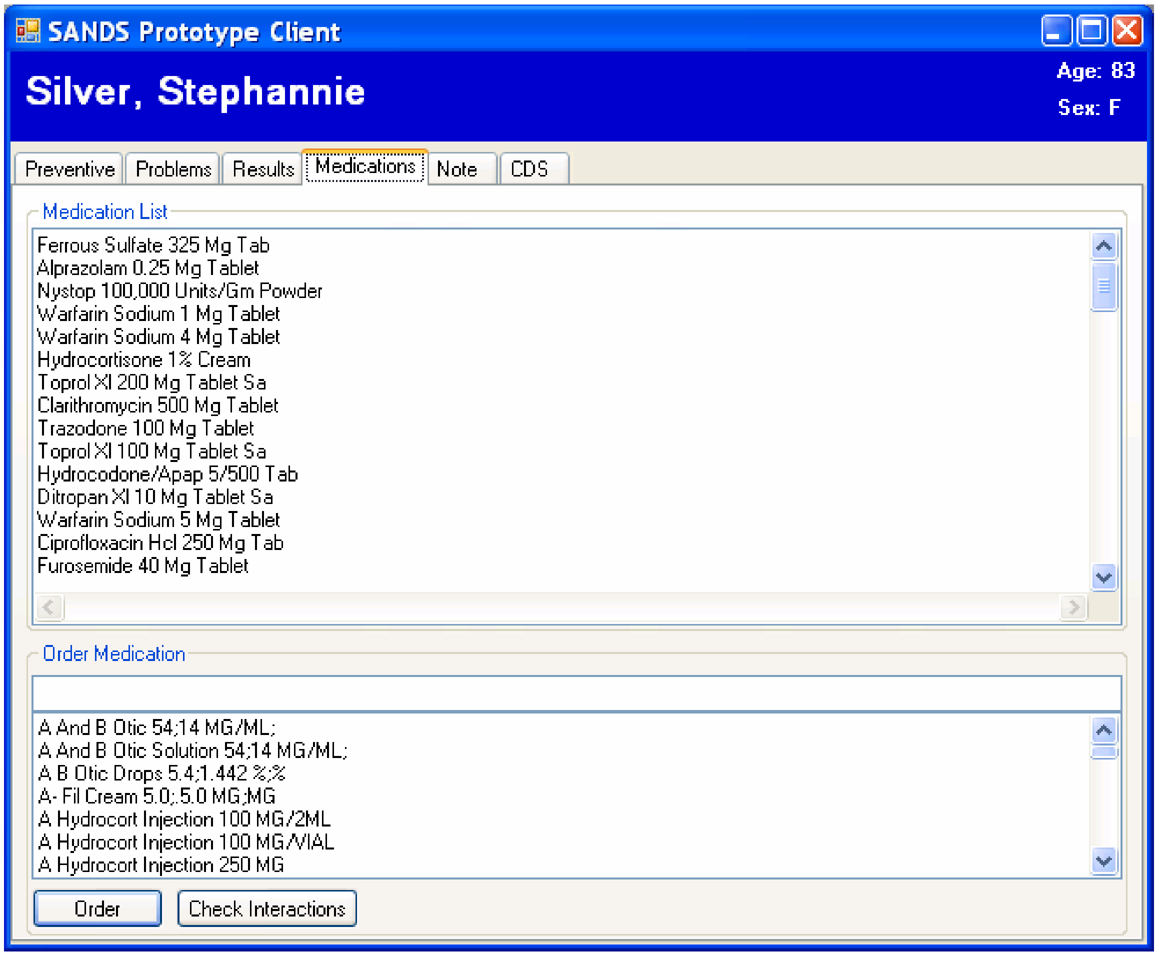
22q partial trisomy syndrome
2-Ketoadipic acidemia
3 beta-Hydroxysteroid dehydrogenase deficiency
3-Methylglutaconic aciduria
5,10-Methylenetetrahydrofolate reductase deficiency
A pattern strabismus
AA amyloidosis

FIGURE 10 - SAMPLE PROBLEM LIST IN THE SANDS PROTOTYPE CLIENT.



Date	Test	Result	Range
8/29/2003	Patient Position	supine (Local Concept: 3354)	
8/29/2003	MAP Auto	115	82-102
8/29/2003	Pulse Auto	66	60-100
8/29/2003	Diastolic BP Auto	68	60-90
8/29/2003	Systolic BP Auto	187	100-160
8/29/2003	Oximetry	93	
8/29/2003	Temp Auto Oral	99.5	97.6-99
8/29/2003	Resp Rate Auto	23	16-20
8/29/2003	Output Shift Vol	257	
8/29/2003	Intake 24H Vol	802	
8/29/2003	Output 24H Vol	756	
8/29/2003	Output Urine Vol/8hrs	257	
8/29/2003	Net I&O 24H	56	
8/29/2003	IN Oral/24H	807	
8/29/2003	OUT Urine/24H	750	
8/29/2003	Intake & Output	-257	
8/29/2003	LV Diam Diastolic	4.7	3.6-5.2
8/29/2003	Height(In)	61	

FIGURE 11 - SAMPLE RESULTS TAB IN THE SANDS PROTOTYPE CLIENT.


SANDS Prototype Client

Silver, Stephannie

Age: 83

Sex: F

Medication List

Ferrous Sulfate 325 Mg Tab
Alprazolam 0.25 Mg Tablet
Nystop 100,000 Units/Gm Powder
Warfarin Sodium 1 Mg Tablet
Warfarin Sodium 4 Mg Tablet
Hydrocortisone 1% Cream
Toprol Xl 200 Mg Tablet Sa
Clarithromycin 500 Mg Tablet
Trazodone 100 Mg Tablet
Toprol Xl 100 Mg Tablet Sa
Hydrocodone/Apap 5/500 Tab
Ditropan Xl 10 Mg Tablet Sa
Warfarin Sodium 5 Mg Tablet
Ciprofloxacin Hcl 250 Mg Tab
Furosemide 40 Mg Tablet

Order Medication

A And B Otic 54;14 MG/ML;
A And B Otic Solution 54;14 MG/ML;
A B Otic Drops 5.4;1.442 %;%
A- Fil Cream 5.0;.5.0 MG;MG
A Hydrocort Injection 100 MG/2ML
A Hydrocort Injection 100 MG/VIAL
A Hydrocort Injection 250 MG

FIGURE 12 - SAMPLE MEDICATION LIST IN THE SANDS PROTOTYPE CLIENT.

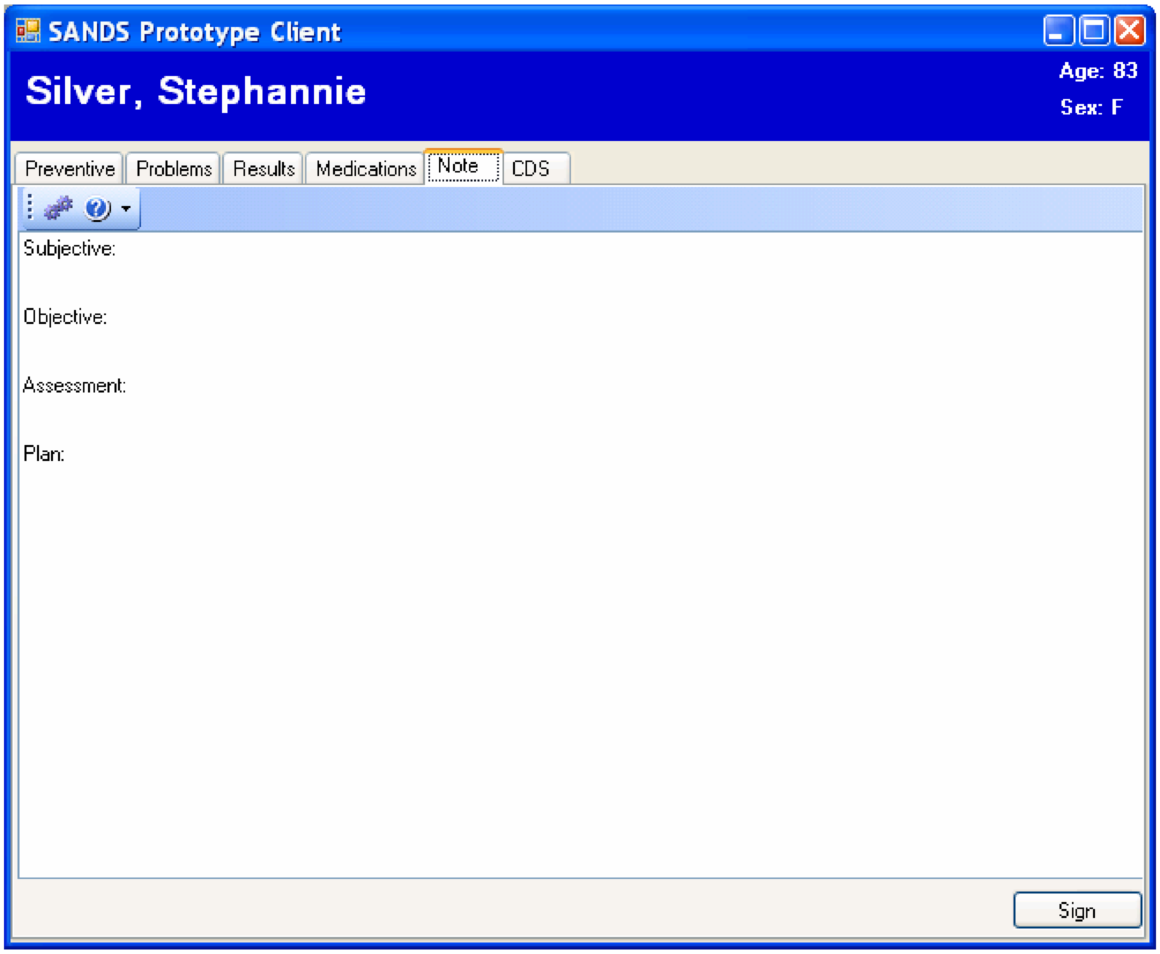


FIGURE 13 - SAMPLE PROBLEM PROGRESS NOTE ENTRY DIALOG IN THE SANDS PROTOTYPE CLIENT.

The other unique feature of the SANDS Prototype Client is, of course, its support of the SANDS architecture.

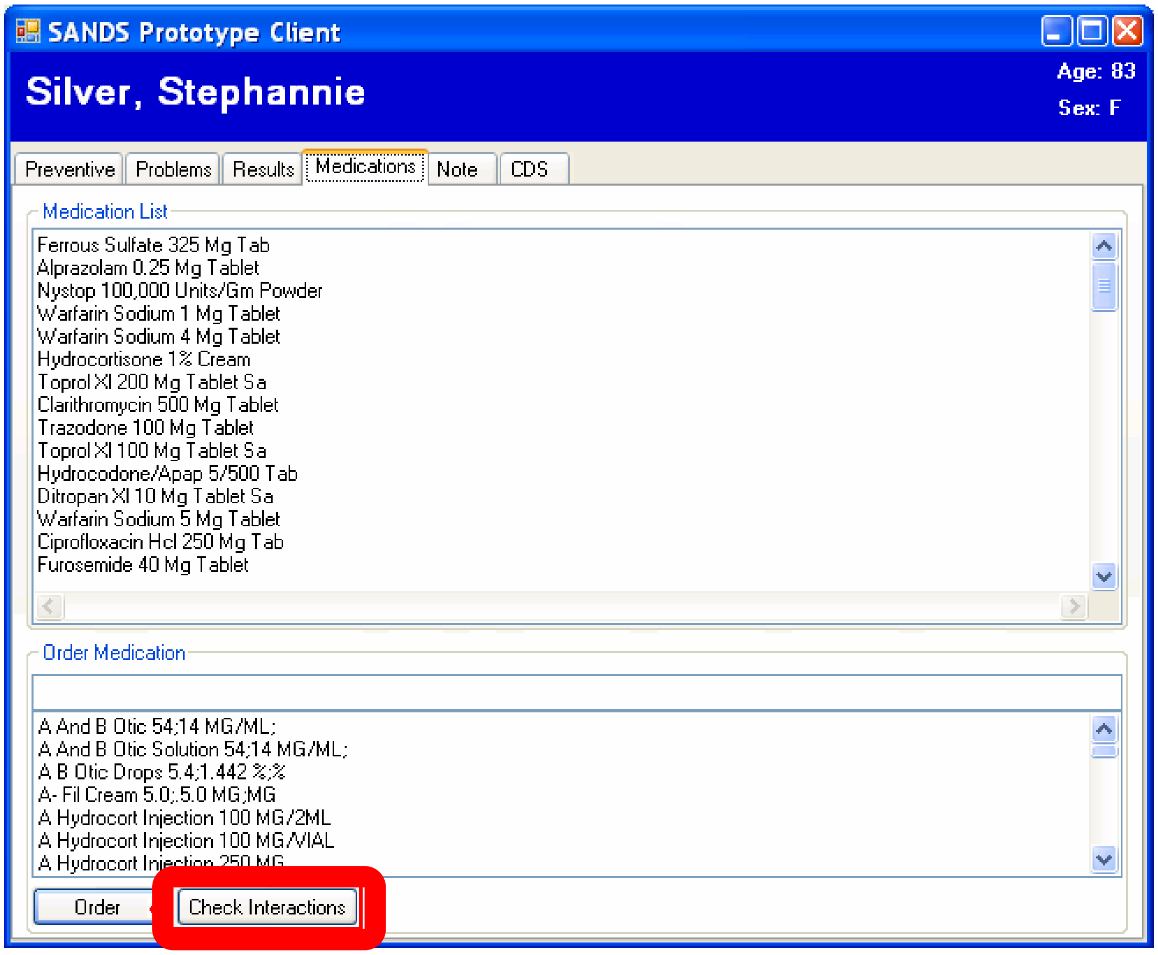
4.7 DRUG INTERACTION CHECKING

The next six sections of the dissertation describe sample use cases developed according to the decision support service definition described earlier in this chapter and in Appendices A an B. These use cases were developed in order to provide real-world validation of the architecture contemplated and the interfaces developed for it.

The first use case is a classic one: drug-drug interaction checking. Many medications, when given in combination, interact, leading to a variety of possible adverse results including direct toxicity, or overactivity or underactivity of one or both of the interacting medications. These interactions are frequent, and can be very dangerous (127-130). However, automated alerting systems can be very effective at reducing both the rate of interacting medication orders, and harmful adverse sequelae (2, 51, 131-133). However, such systems are not universally adopted, and, where used, are frequently part of pharmacy computer systems rather than ordering systems, limiting their effectiveness at the point of care.

It would certainly be desirable to increase the adoption of effective drug-drug interaction alerting in ordering systems and several commercial products and databases of drug-drug interactions are available (134). However, integrating these databases into clinical systems can frequently be difficult, and in many cases, vendors of clinical systems have relationships with a single database provider limiting users of these clinical systems to only that provider of drug information. In this use case, I developed a service for a drug interaction checking based on a commercial drug information database developed by Lexi-Comp, Inc. The Lexi-Comp database was chosen because Lexi-Comp offers a pre-existing web service interface, which could be readily adapted to the standardized interface developed here.

The SANDS prototype client supports calling the drug interaction service in two modes: first, it can be called when the user clicks the "Check Drug Interactions" button, highlighted in Figure 14. This button checks the patient's current consolidated medication list for drug interactions. Second, the service can be called when a new medication is ordered, to check for interactions between that medication and the rest of the patient's medication list.

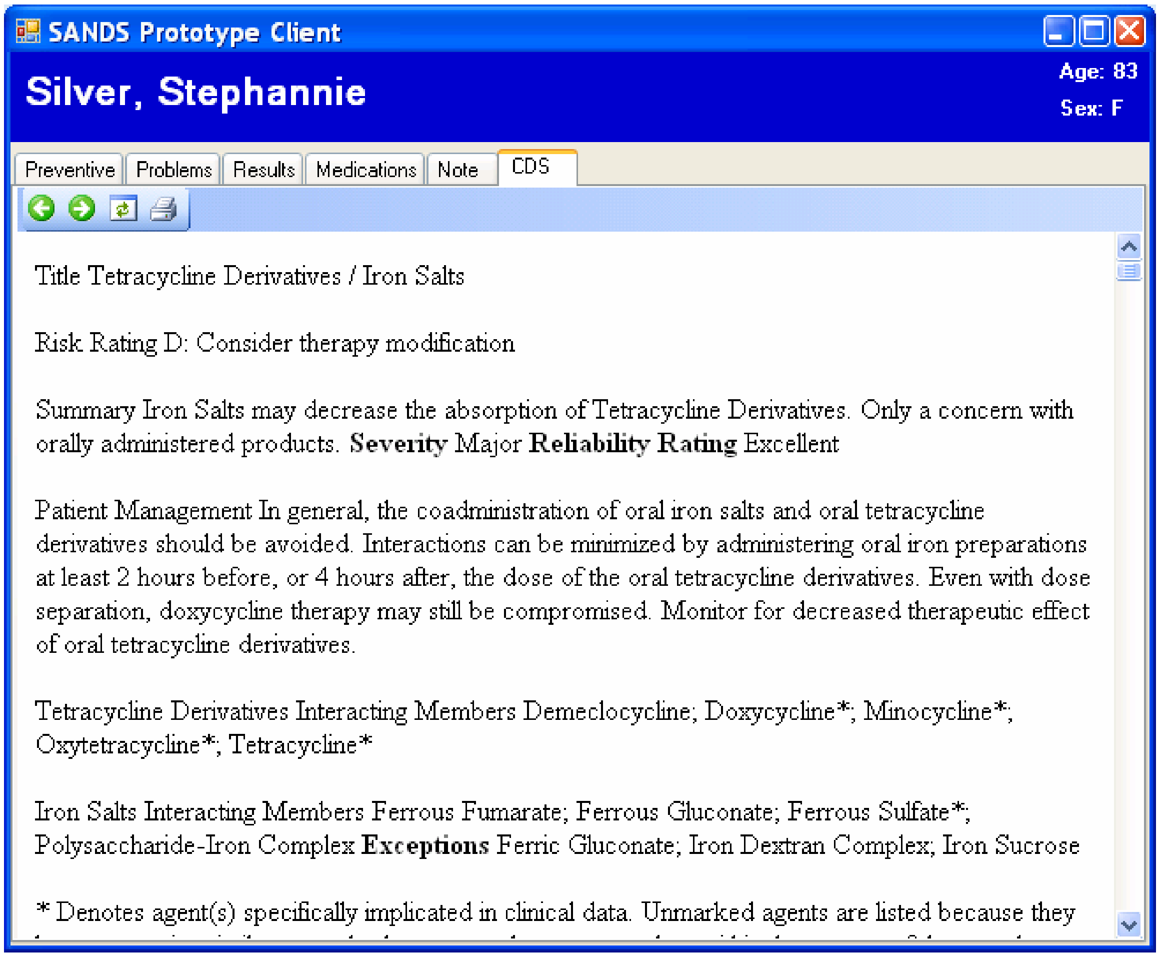


Ferrous Sulfate 325 Mg Tab
Alprazolam 0.25 Mg Tablet
Nystop 100,000 Units/Gm Powder
Warfarin Sodium 1 Mg Tablet
Warfarin Sodium 4 Mg Tablet
Hydrocortisone 1% Cream
Toprol Xl 200 Mg Tablet Sa
Clarithromycin 500 Mg Tablet
Trazodone 100 Mg Tablet
Toprol Xl 100 Mg Tablet Sa
Hydrocodone/Apap 5/500 Tab
Ditropan Xl 10 Mg Tablet Sa
Warfarin Sodium 5 Mg Tablet
Ciprofloxacin Hcl 250 Mg Tab
Furosemide 40 Mg Tablet

A And B Otic 54;14 MG/ML;
A And B Otic Solution 54;14 MG/ML;
A B Otic Drops 5.4;1.442 %;%
A- Fil Cream 5.0;.5.0 MG;MG
A Hydrocort Injection 100 MG/2ML
A Hydrocort Injection 100 MG/VIAL
A Hydrocort Injection 250 MG

FIGURE 14 - THE CHECK INTERACTIONS BUTTON IN THE SANDS PROTOTYPE CLIENT.

Consider a sample case where the patient is taking both doxycycline (a tetracycline-derived antibiotic) and ferrous sulfate (an iron salt used for iron supplementation in anemic patients). Ferrous sulfate reduces the absorption of doxycline, potentially to a sub-therapeutic level (135). When the user clicks the "Check Interactions" button, the drug interaction compares all of the medications on the paient's medication list and generates an alert. The alert contains a link to a monograph which describes the interactions. The SANDS Prototype Client then displays the monograph to the user directly inside of the EHR, as shown in Figure 15.



Title Tetracycline Derivatives / Iron Salts

Risk Rating D: Consider therapy modification

Summary Iron Salts may decrease the absorption of Tetracycline Derivatives. Only a concern with orally administered products. **Severity** Major **Reliability Rating** Excellent

Patient Management In general, the coadministration of oral iron salts and oral tetracycline derivatives should be avoided. Interactions can be minimized by administering oral iron preparations at least 2 hours before, or 4 hours after, the dose of the oral tetracycline derivatives. Even with dose separation, doxycycline therapy may still be compromised. Monitor for decreased therapeutic effect of oral tetracycline derivatives.

Tetracycline Derivatives Interacting Members Demeclocycline; Doxycycline*; Minocycline*; Oxytetracycline*; Tetracycline*

Iron Salts Interacting Members Ferrous Fumarate; Ferrous Gluconate; Ferrous Sulfate*; Polysaccharide-Iron Complex **Exceptions** Ferric Gluconate; Iron Dextran Complex; Iron Sucrose

* Denotes agent(s) specifically implicated in clinical data. Unmarked agents are listed because they

FIGURE 15 - A DRUG INTERACTION MONOGRAPH, SHOWING AN INTERACTION BETWEEN TETRACYCLINE DERIVATIES AND IRON SALTS. (*MONOGRAPH TEXT PROVIDED BY LEXI-COMP, INC.*).

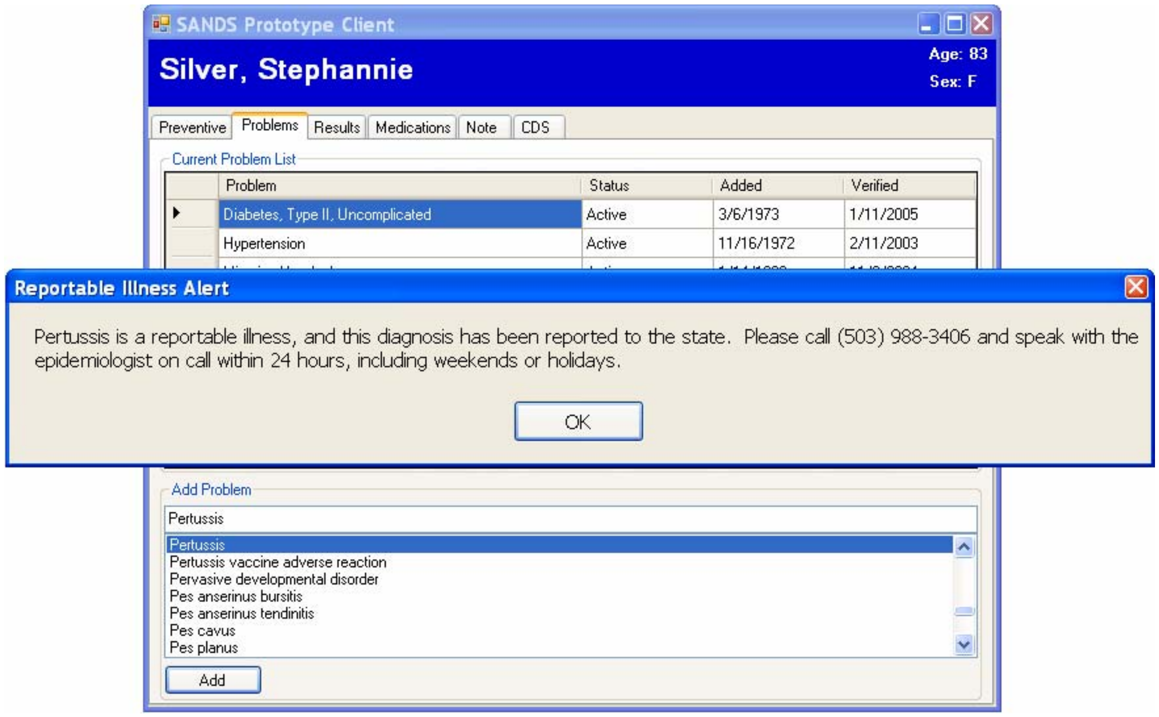
Integrating the drug-drug interaction service with the clinical system using the SANDS architecture was achieved quite easily, and the intervention seems effective. As discussed previously, it is well-known that effective drug-drug interaction alerting systems can reduce adverse drug events. The only major issue encountered in the integration process relates to the threshold for alerts. It is widely thought that commercial drug information databases have too low a threshold for alerts (i.e. that they alert too often, and that many alerts they provide are not clinically relevant) (134, 136-138), and that appeared to be the case here. The Lexi-Comp drug interaction service has a severity grading system, but alerts graded C, D or X (those alerts that Lexi-Comp classifies as needing human intervention) are extremely common. For example, one sample patient's consolidated medication list yielded 750 alerts, 685 of which were graded C, D or X. While some of the interactions were clinically significant (such as the example presented above), it would be impractical for a clinical to review 685 separate alerts. Before a drug-drug interaction service could be deployed for real-world use, it would be important to thoughtfully pare down the interaction database, a process already underway at some advanced sites (139, 140).

4.8 SYNDROMIC SURVEILLANCE

The next use case I developed was for syndromic surveillance and reporting of reportable diseases to public health authorities. Unlike the previously presented use cases, there was no pre-existing service for this function. Instead I developed the decision support system *de novo* based on published documents from the Oregon Health Department. Like most states, epidemiologists at the Oregon Health Department track the spread of a variety of infectious disease. One key element of this tracking program is mandatory reporting of diseases. Some of this reporting is carried out by medical laboratories which are obligated to report positive tests for diseases in reportable groups. However, certain diseases, such as pertussis, are often diagnosed solely on clinical factors, so the responsibility for reporting the disease falls on the provider that made the diagnosis. Although data about how often reportable diseases are actually reported are sparse, it is widely believed that there is significant underreporting of diseases, largely due to providers who either are not aware of the rules or who do not have the time to comply with them. To support both providers and public health authorities I developed a reporting system which takes all new diagnoses, at the time they are entered by providers, and runs them through a filter to determine whether or not they represent reportable diseases. This use case makes interesting use of the SNOMED terminology. Problems reported to the decision support system are reported according to SNOMED concept ID's, which provides a good level of specificity. In many cases, determining whether a disease is reportable or not is a simple matter of comparing the concept ID being reported to a list of the concept ID's for reportable diseases. However, in certain cases, the analysis is more complex. Consider, for example infection by the organism *Nesseria gonorrhoeae*, which is reportable under Oregon law. While a SNOMED concept ID (15628003) exists for such an infection, there are actually 90 clinical concepts which descend from this term, all of which would be considered reportable in Oregon. However because SNOMED has a robust semantic structure it is unnecessary to identify, *a priori*, all of these concepts. Instead, the rule is defined to include SNOMED concept 15628003, and all of its descendents. If, in the future, new problems are added under concept 15628003, any decision support system which uses the semantic structures of SNOMED will automatically have the benefit of these updates, without needing to be modified.

In the current embodiment of this prototype, whenever a new problem is entered into the prototype electronic health record which is reportable under Oregon law a pop-up is given to the user based on a database which describes the reporting requirements for each disease. The alert informs the provider whether or not they need to contact the state epidemiologist directly or whether the automatic notification already received is sufficient. If it is necessary to contact the epidemiologist, the system provides detailed information to the clinician about the phone number to call and how long he or she has to report the disease. This is important because some diseases require immediate consultation with the epidemiologist while others can be reported days later.

Figure 16 shows an example of this feature. In this case, a diagnosis of pertussis was entered for a patient who lives in Multnomah County, Oregon. In Oregon pertussis is a reportable disease, so an alert is triggered. Because the patient lives in Multnomah County, the reporting number for the appropriate health department is also provided.


SANDS Prototype Client

Silver, Stephannie

Age: 83
Sex: F

Preventive | Problems | Results | Medications | Note | CDS

Current Problem List

Problem	Status	Added	Verified
Diabetes, Type II, Uncomplicated	Active	3/6/1973	1/11/2005
Hypertension	Active	11/16/1972	2/11/2003

Reportable Illness Alert

Pertussis is a reportable illness, and this diagnosis has been reported to the state. Please call (503) 988-3406 and speak with the epidemiologist on call within 24 hours, including weekends or holidays.

OK

Add Problem

Pertussis

Pertussis
Pertussis vaccine adverse reaction
Pervasive developmental disorder
Pes anserinus bursitis
Pes anserinus tendinitis
Pes cavus
Pes planus

Add

FIGURE 16 – AN ALERT REMINDING THE CLINICIAN THAT PERTUSSIS IS A REPORTABLE ILLNESS IN OREGON, AND PROVIDING INSTRUCTIONS FOR HOW TO REPORT IT.

This service stands out amongst the others in that it not only provides an interface to an electronic health record, but also provides a separate interface to public health management systems. In this case, an epidemiologist with proper authorization can query the system for a list of all cases that match certain query parameters. For example, an epidemiologist might query the database to see where cases of pertussis were reported within the Portland metropolitan area over the past seven days. The current prototype system then links to Google Earth, a 3-D map visualization, program to provide a geospatial representation of the cases in the database. This requires that the disease reporting service, in turn, consults a geocoder service. Geocoding is the process of taking an address as might be stored in a patient's electronic health record and converting that address to a spatial location, usually the latitude and longitude at that point. Figure 17 shows how these services

interact.

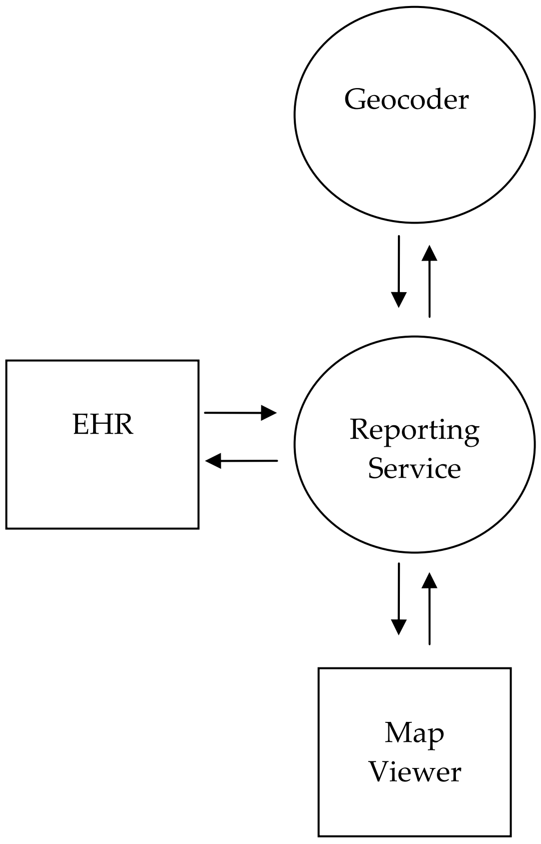


FIGURE 17 – SCHEMATIC SHOWING RELATIONSHIP BETWEEN THE REPORTING SERVICE, AND THE OTHER SYSTEMS AND SERVICES IT CONNECTS TO.

In the typical interaction between the electronic health record and the reporting service, the EHR would send a problem-added invocation to the reporting service. If necessary the reporting service would respond to the EHR with a notify intervention, explaining that the problem most recently added represents a reportable disease.

This use case demonstrates three special aspects of this architecture. First, it shows the ability of a single service to provide decision support to two different kinds of clinical systems. There are many use cases where this could be helpful: for example, a drug allergy warning service which interfaces with both an electronic health record and a personal health record.

Second this use case demonstrates service composition: the reporting service contacts another service, the geocoder, to complete its work. Many cases of decision support will use service composition. For example, a therapeutic reminder system might first query an allergy system to determine whether or not the therapy that it is about to suggest is contraindicated due to a patient allergy.

Third, this use case demonstrates the ability to modularly replace services. In this project and another related project, performed for the Oregon State Health Department, six different geocoding services were reviewed. Each of the services have advantages and disadvantages, with their primary differentiation in the quality of their mapping and their cost. A number of free services are available, provided by Google, Yahoo and others. These services generally use free but lower quality maps (such as the Census Tiger Line File). Paid services generally use higher quality maps, such as those from TeleAtlas and NavTeq. Because one of the goals of this project is to showcase free software and free services, and because accuracy down to be house level is not especially important for the public health use case, I chose to use the Google geocoder service. The service is free, and provides reasonably high quality matches with very high system availability. However, for their purposes the state chose to use a higher quality and more accurate geocoding service, which requires payment. The ability to freely choose from a variety of decision support services makes development, testing and maintenance of software much easier, and it is hoped that one day the robust and competitive market we saw for geocoding services might be replicated for decision support services.

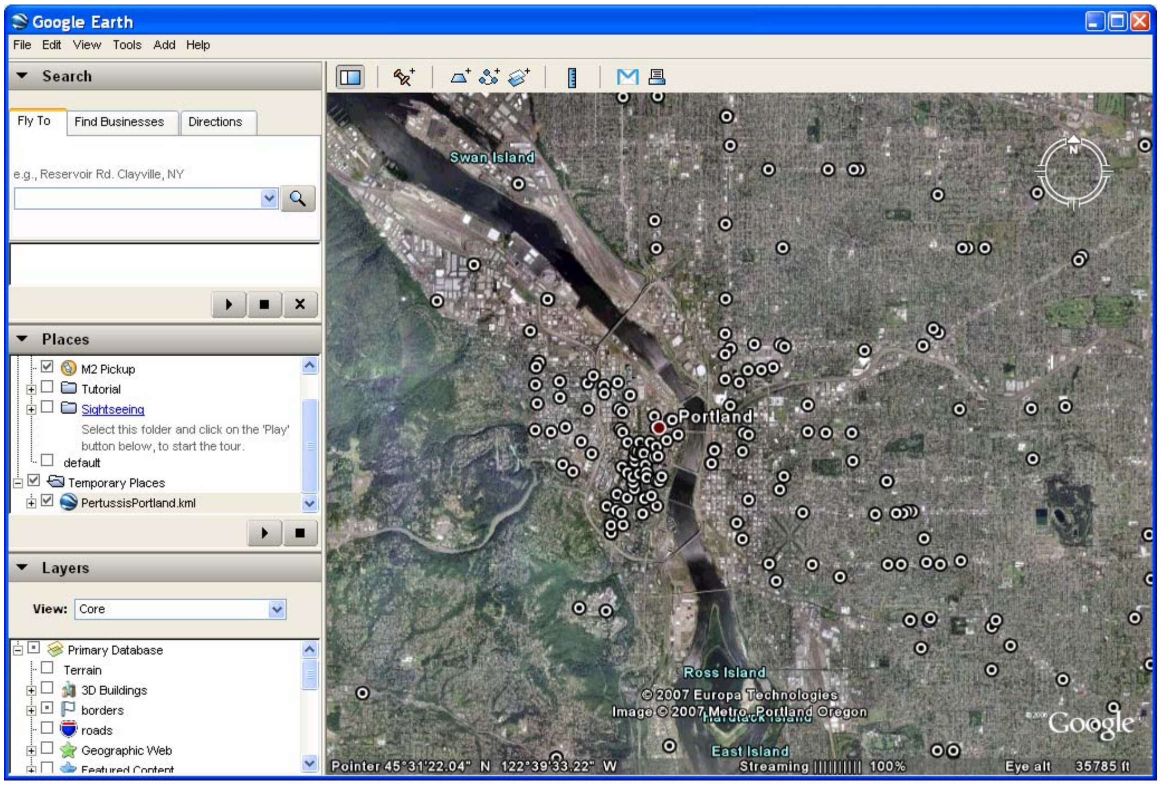


FIGURE 18 – SIMULATED PERTUSSIS CASES FROM THE SYNDROMIC SURVEILLANCE / REPORTABLE DISEASE SERVICE, AS VIEWED IN GOOGLE EARTH.

4.9 DIAGNOSTIC DECISION SUPPORT

The next use case is a diagnostic decision support system. Diagnostic decision support systems have been used for the past 30 years. Two of the most ambitious systems were the INTERNIST system (16) developed by Randoph Miller and DXPlain (10) developed by Octo Barnett at Massachusetts General Hospital. Both systems are described in the background section of this dissertation. It is important to note, however, that neither of these systems achieved widespread clinical use in the real world, largely because they were very difficult to integrate into clinical workflows. However, recently, novel approaches to diagnostic decision support systems have been pursued, and a number of new diagnostic decision support systems are available which make it easier to integrate such decision support into clinical workflows.

In my prototype architecture, I built an interface between the Isabel diagnostic decision support system (Isabel Healthcare, Reston, Virginia) and my prototype electronic health record. The Isabel system uses rudimentary natural language processing techniques to provide clinical decision support instead of using a comprehensive knowledge base of medicine as previous systems have. Isabel uses an automated text mining system to review medical literature and medical textbooks. It then compares the structure and semantic content of these books with a narrative report entered by the user. In this case the interface with Isabel sends the subjective and objective components of a progress note, as entered in my prototype electronic health record system, to Isabel for processing.

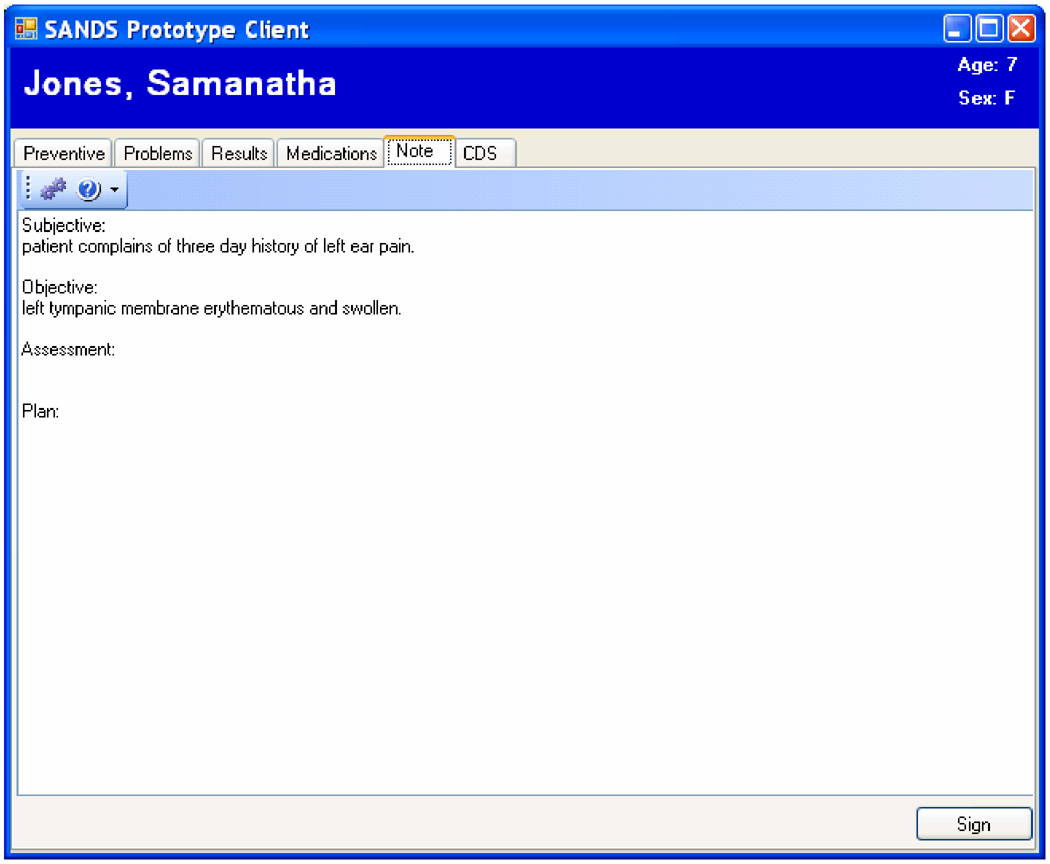
Isabel already provided a simple web service interface to enable their product to be integrated with EMR systems. This interface was proprietary, however, so I developed a wrapper to map the proprietary interface to the SANDS architecture standard interface. In many respects this interface was the easiest that I created because Isabel uses free text and has its own natural language processing engine and its own thesaurus. As such, it was not necessary to provide any coded data to Isabel so there were no vocabulary standards issues to resolve.

Consider a simple case of a seven-year-old child with left ear pain. The provider has entered the subjective and objective components of his note:

SUBJECTIVE: PATIENT COMPLAINS OF THREE DAY HISTORY OF LEFT EAR PAIN.

OBJECTIVE: LEFT TYMPANIC MEMBRANE ERYTHEMATOUS AND SWOLLEN.

The entry of this note into the SANDS prototype client is shown in Figure 19.



Subjective:
patient complains of three day history of left ear pain.

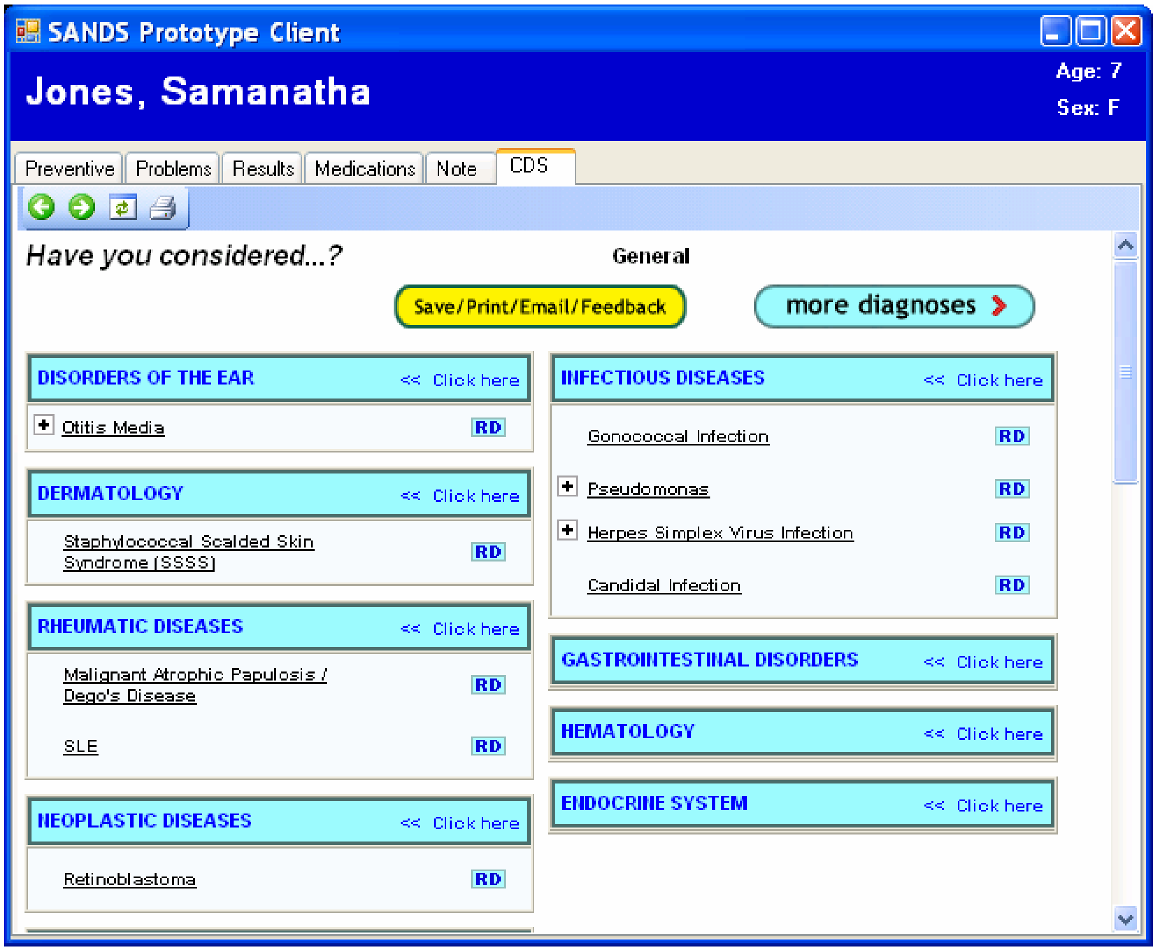
Objective:
left tympanic membrane erythematous and swollen.

Assessment:

Plan:

FIGURE 19 – A PROGRESS NOTE ENTERED IN THE SANDS PROTOTYPE CLIENT.

Now, before proceeding to the assessment and plan, the provider activates the Isabel decision support system over the prototype architecture by clicking a button in his electronic medical record. This triggers the electronic medical record system to send a standard message to the Isabel decision support system containing the text of the note entered so far by the practitioner as well as a pointer to the patient's electronic medical record and the national health information network. The diagnostic decision support system uses this note as well as other background information such as demographics retrieved from the NHIN. Figure 20 shows Isabel's response.


SANDS Prototype Client

Jones, Samanatha

Age: 7
Sex: F

Have you considered...?

General

DISORDERS OF THE EAR << Click here
Otitis Media RD

DERMATOLOGY << Click here
Staphylococcal Scalded Skin Syndrome (SSSS) RD

RHEUMATIC DISEASES << Click here
Malignant Atrophic Papulosis / Dego's Disease RD
SLE RD

NEOPLASTIC DISEASES << Click here
Retinoblastoma RD

INFECTIOUS DISEASES << Click here
Gonococcal Infection RD
Pseudomonas RD
Herpes Simplex Virus Infection RD
Candidal Infection RD

GASTROINTESTINAL DISORDERS << Click here

HEMATOLOGY << Click here

ENDOCRINE SYSTEM << Click here

FIGURE 20 – DIAGNOSTIC DECISION SUPPORT RESPONSE FROM ISABEL, AS DISPLAYED IN THE SANDS CLIENT.

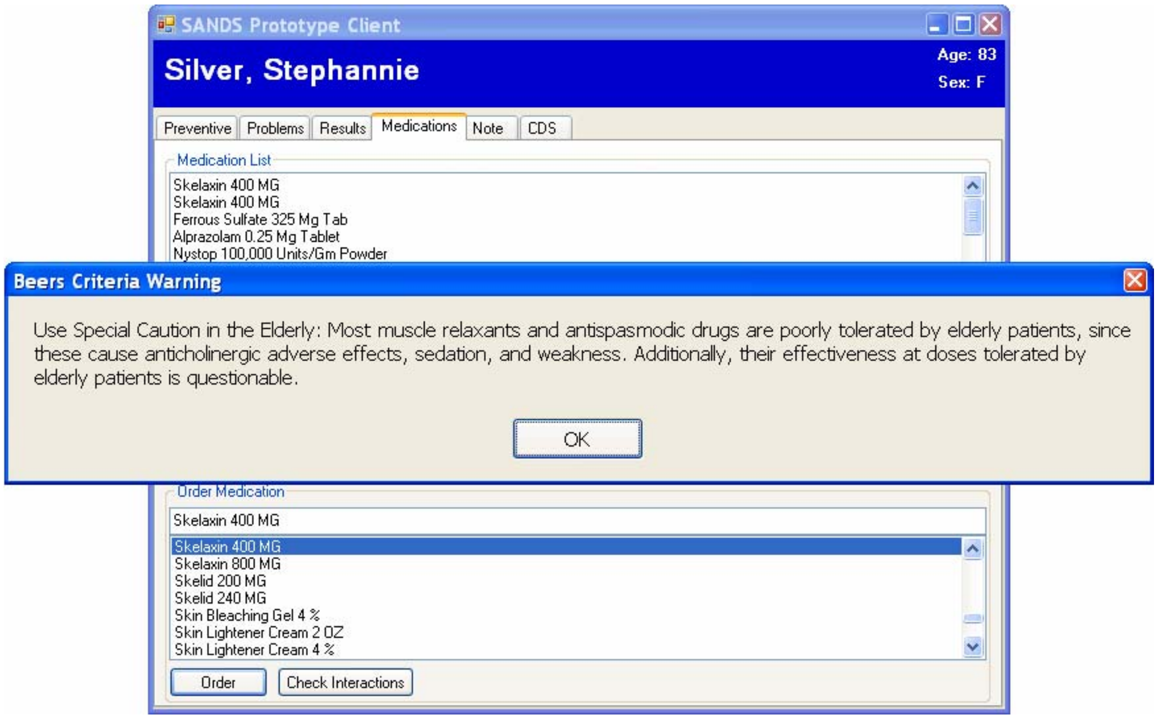
Based on its knowledge base, Isabel provides a differential containing nine diagnoses across five systems. In this case, the first diagnosis proposed by Isabel is the correct one. This is not always the case as Isabel does not attempt to order its diagnosis list. Instead, Isabel's goal is to provide the correct diagnosis within the first screen. Although this case may appear trivial to a human, it is still impressive given the past challenges that diagnostic decision support systems have faced. Published evidence suggests that Isabel provides good accuracy, even in very complex cases (141-144).

One difficulty encountered in integrating Isabel into the decision support workflow was the fact that, in current versions, Isabel provides its differential diagnosis results in an unstructured format. This precluded developing a feature allowing the user to click a diagnosis in Isabel and have it automatically added to the progress note or problem list. In future work, we hope to map Isabel's diagnoses to SNOMED so they can be more readily integrated into clinical systems. Were such codes already available, we would use the `addProblem` response type so that a compliant electronic health record system could automatically add the chosen diagnosis to the problem list.

4.10 INAPPROPRIATE PRESCRIBING IN OLDER ADULTS

The next use case is a drug-therapy decision support system designed to assist clinicians in prescribing medications for older adults. Certain medications are metabolized differently in older adults than in younger people, and it is important to take this into account when prescribing. To this end, an expert consensus panel developed guidelines which specify drugs which are either unsafe in older adults, or which require dose adjustment (145). This guidance is referred to as the Beers criteria, after Dr. M. H. Beers, a past editor of the Merck Manual, and author of an early list of medications whose use is contra-indicted in older adults. Although this guidance is available, it is well-known that such drugs are still frequently used in older adults, even when better alternatives are available (146).

Because no existing Beers criteria decision support service was available, I developed one from scratch, based on the guidance published in Archives of Internal Medicine (145). Figure 21 shows a sample alert raised by the service, in response to an order for Skelaxin (metaxalone, a skeletal muscle relaxant) in an 83 year old female.



Skelaxin 400 MG
Skelaxin 400 MG
Ferrous Sulfate 325 Mg Tab
Alprazolam 0.25 Mg Tablet
Nystop 100,000 Units/Gm Powder

Use Special Caution in the Elderly: Most muscle relaxants and antispasmodic drugs are poorly tolerated by elderly patients, since these cause anticholinergic adverse effects, sedation, and weakness. Additionally, their effectiveness at doses tolerated by elderly patients is questionable.

Skelaxin 400 MG

Skelaxin 400 MG
Skelaxin 800 MG
Skelid 200 MG
Skelid 240 MG
Skin Bleaching Gel 4 %
Skin Lightener Cream 2 OZ
Skin Lightener Cream 4 %

FIGURE 21 - AN ALERT BASED ON THE BEERS CRITERIA.

The inference pattern of this service is simple rule-based reasoning. The service maintains a memory-resident associative array (or dictionary), which binds medications to alert text. When the service is invoked for a new medication order, it fetches the patient's age. If the patient is 65 or younger, no alert can be returned, and the logic terminates. If the patient is over 65, the array is queried for the medication being ordered (an O(1) operation). If the medication is not found, no alert is returned, and the logic terminates. However, if an alert is found, a notify response containing the appropriate alert text is generated (as described in section 4.3) and returned to the requesting clinical system.

One interesting facet of this use case is the method by which its content was developed. While the Beers criteria was developed by an expert consensus panel, it has been formatted for use in a clinical information system through a wiki (the Clinfowiki: http://www.clinfowiki.org) using what Benkler has called "commons-based peer production" (147). In this model, collaborators (including the author of this dissertation) have worked to refine the criteria, add alert text and put the data into machine-readable format using wiki software (148). This method seems to work fairly well, and the alerts are now available online (http://www.clinfowiki.org/wiki/index.php?title=Tab-separated_file_of_Beers_criteria_alerts), under the GNU Free Documentation License.

4.11 INFORMATION AT THE POINT OF CARE

During the course of caring for patients, clinicians have a variety of information needs. One oft-cited study conducted by Paul Gorman and Mark Helfand in 1995 found that clinicians have, on average, one question for every two patients they see, but that they only pursue about 30% of these questions, and that they use the computer infrequently (149). More recent studies have found similar results, and while use of computers for point of care information needs is increasing, print sources still dominate (150-152). In many cases, print sources may be the most convenient, but electronic sources have some critical advantages: they can be updated more frequently, they can be accessed by multiple people, and they support searching. It is known that, given the right tools, clinicians are willing to use electronic reference sources (152, 153), and that good, usable electronic references sources exist (154).

However, it has been difficult to connect clinicians to these electronic resources for two reasons: first, clinicians who are using an EHR and who wish to consult such a source must generally open a web browser outside of the EHR, access the reference source and log-in, and then must consult the source, while possibly switching back and forth between the EHR and the reference source. These context switches impose a high cognitive burden. Second, clinicians frequently have contextual questions – for example, they are ordering a drug, and want information on appropriate dosing. In many cases, they have already created this context in their EHR by, for example, starting an order, and then they are forced to re-create the same context in the separate information tool. Various attempts, such as the Infobutton standard, have been made to more closely integrate information resources with clinical information systems, and early results are at least somewhat encouraging (155, 156).

In this use case, I developed a decision support system for providing context-specific information to clinicians at the point of care. I chose not to use the Infobutton standard for my interface because of various licensing and patent encumbrances on the standard (157). I used two information sources for my service: UpToDate and Google Co-op. UpToDate is a commercial source of evidence summaries for a variety of conditions that might be encountered in primary care. It is widely used, and very popular with clinicians (154). It is also very expensive . Google Co-op, on the other hand, is free. It is a health search engine based on the main Google index, but supplemented with tags added by 26 trusted contributors, such as the Health on the Net Foundation, the Mayo Clinic, Kaiser Permanente, the National Library of Medicine and Harvard Medical School. These contributors add tags to articles, describing the type of information they contain, thus helping to improve the quality of search results for medical conditions and drugs. Both of these sources use free-text search, and neither of their indexes are linked to any structured vocabulary, so direct, free-text search terms are passed to them through the decision support interface.

This decision support service is accessed by context menus in the SANDS prototype client. Figure 22 shows a sample context menu for information about a disease. This menu allows the user to choose whether they want to query UpToDate or Google. If the user wants to query Google, there are options to narrow the search to treatment, symptoms, tests / diagnosis, causes / risk factors, practice guidelines, patient handouts and clinical trials. These correspond to disease information tags in the Google Co-op knowledge base.

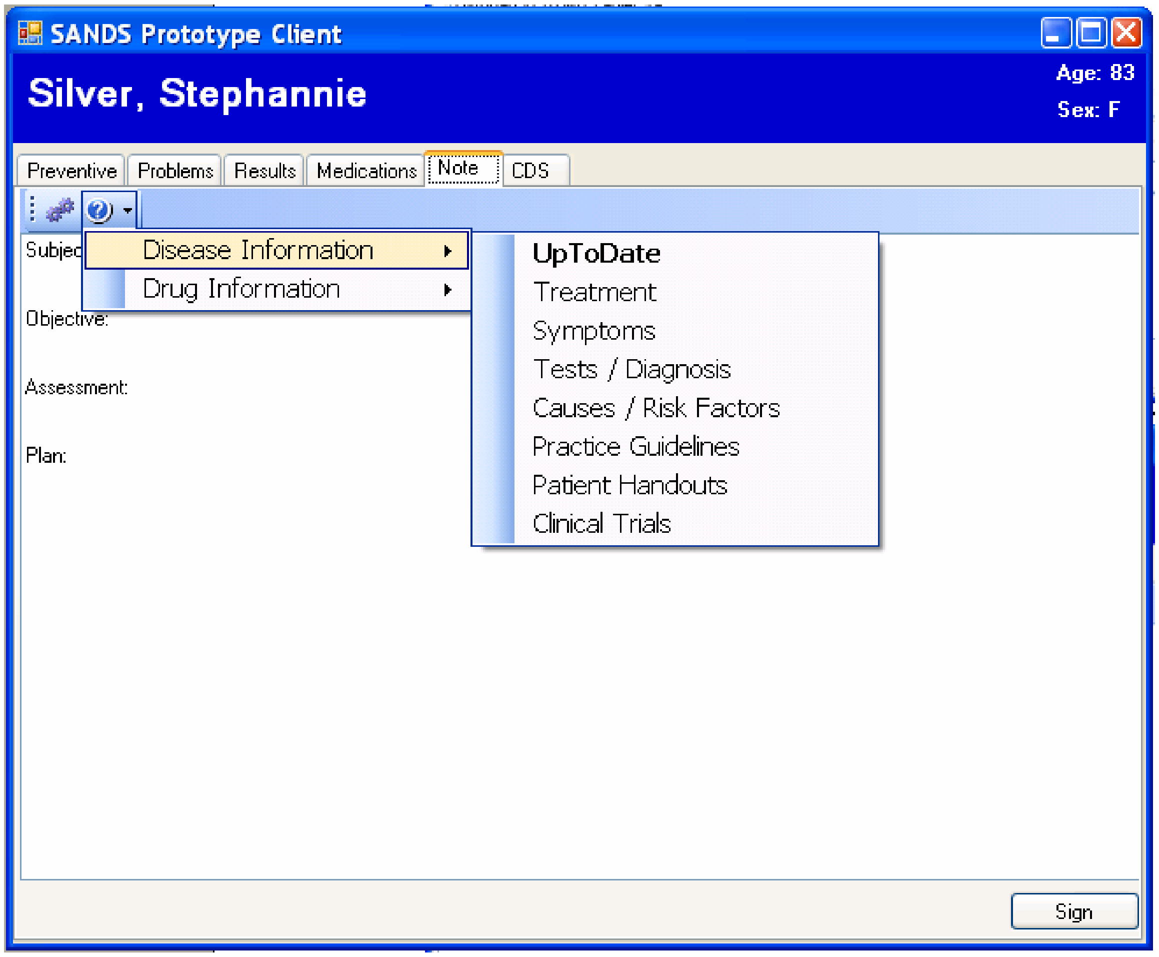
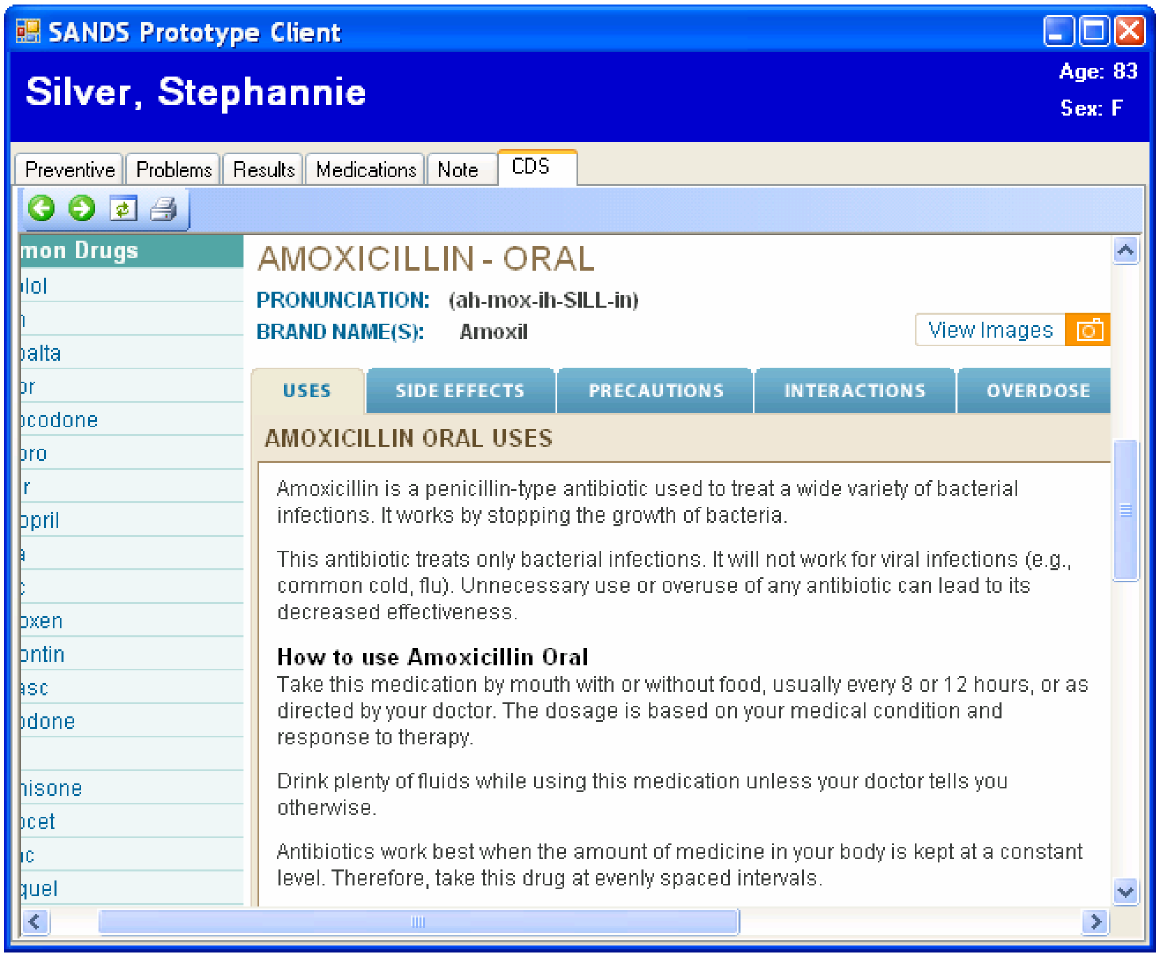


FIGURE 22 - POINT OF CARE INFORMATION CONTEXT MENU IN THE SANDS PROTOTYPE CLIENT.

The same service is called regardless of what category is selected, with the information category selected by the user passed in to the decision support service as a parameter. In a typical use case, a clinical system analyst at a hospital would use the user interface design toolkit provided by their clinical system vendor to develop a custom menu, where each item was a hook into an information service. The clinical system designer would customize the menu items available based on the information sources that the hospital subscribes to and the information needs of clinicians at that hospital.

Figure 23 shows the result of a user request for information from UpToDate on otitis media. In this case, the user is taken directly to an index of UpToDate's knowledge content on otitis media (not to the front page of UpToDate), and the content is provided inside of the clinical system, without needing to switch to a web browser. Print capability is supported, so the user can print copies of information for patients.



AMOXICILLIN - ORAL

PRONUNCIATION: (ah-mox-ih-SILL-in)

BRAND NAME(S): Amoxil

AMOXICILLIN ORAL USES

Amoxicillin is a penicillin-type antibiotic used to treat a wide variety of bacterial infections. It works by stopping the growth of bacteria.

This antibiotic treats only bacterial infections. It will not work for viral infections (e.g., common cold, flu). Unnecessary use or overuse of any antibiotic can lead to its decreased effectiveness.

How to use Amoxicillin Oral

Take this medication by mouth with or without food, usually every 8 or 12 hours, or as directed by your doctor. The dosage is based on your medical condition and response to therapy.

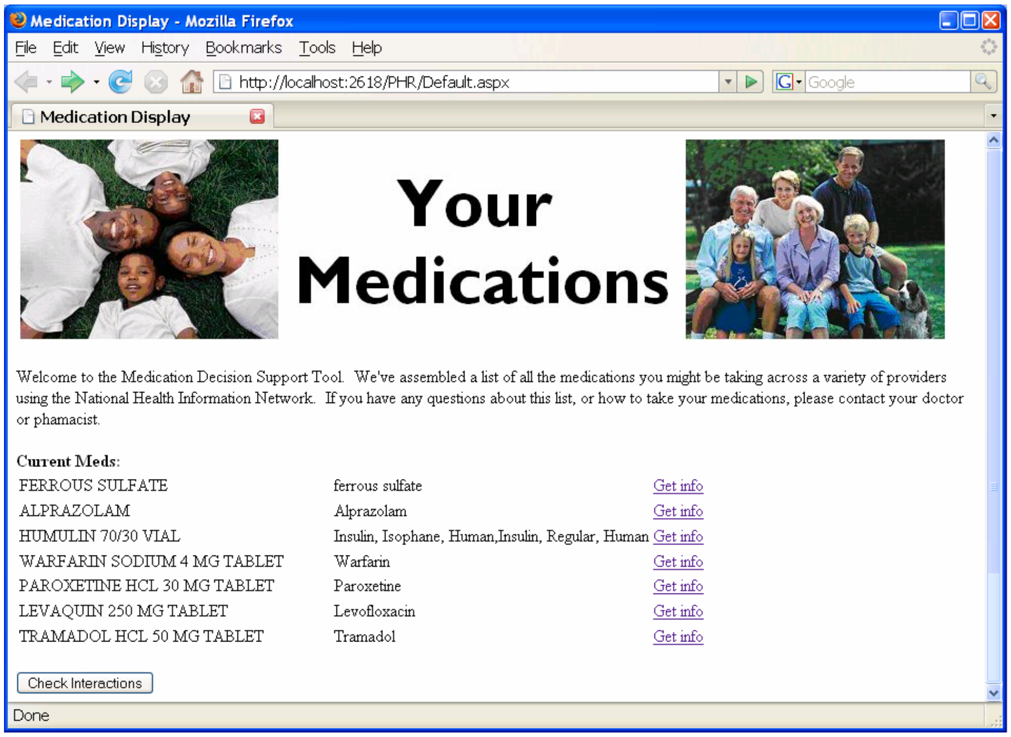
Drink plenty of fluids while using this medication unless your doctor tells you otherwise.

Antibiotics work best when the amount of medicine in your body is kept at a constant level. Therefore, take this drug at evenly spaced intervals.

FIGURE 23 - PATIENT HANDOUT ON AMOXICILLIN.

4.12 A SIMPLE PERSONAL HEALTH RECORD

The final use case that I developed was a simple personal health record, shown in Figure 24. It is a web-based application that a patient would access. It provides a read-only view into the patient's medication list, as stored in the NHIN, allowing the patient to see what medications are active across a number of providers. The personal health record also provides drug-interaction checking and links to information resources targeted at patients. These services are actually identical to the services provided through the EHR and discussed in sections 4.7 and 4.11. This is one of the key advantages of the SANDS architecture – because decision support content is exposed as services, the content can be reused across different end-user applications (the EHR, as described in sections 4.7 and 4.11) and the PHR as described here.


Medication Display - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:2618/PHR/Default.aspx
Google
Medication Display
Your Medications
Check Interactions
Done

Welcome to the Medication Decision Support Tool. We've assembled a list of all the medications you might be taking across a variety of providers using the National Health Information Network. If you have any questions about this list, or how to take your medications, please contact your doctor or phamacist.

Current Meds:

FERROUS SULFATE	ferrous sulfate	Get info
ALPRAZOLAM	Alprazolam	Get info
HUMULIN 70/30 VIAL	Insulin, Isophane, Human,Insulin, Regular, Human	Get info
WARFARIN SODIUM 4 MG TABLET	Warfarin	Get info
PAROXETINE HCL 30 MG TABLET	Paroxetine	Get info
LEVAQUIN 250 MG TABLET	Levofloxacin	Get info
TRAMADOL HCL 50 MG TABLET	Tramadol	Get info

FIGURE 24 - PERSONAL HEALTH RECORD USE CASE.

4.13 NOTES ON LOCAL MIRRORING OF SERVICES

Service performance, reachability and uptime is discussed in detail in Chapter 6 of this dissertation. Overall, though, the performance of these services proved to be quite good, and the overhead added by the SANDS architecture was minimal. That said, it is recognized that some users might prefer to run services locally. In the case of locally-developed services this is natural, however, there is also likely to be demand for mirroring services. As such, a mechanism for mirroring services was developed.

The mirroring mechanism is based on HTTP, so each service that supports mirroring must provide a mirror URL. All SANDS decision support services, regardless of whether or not they support mirroring, should provide a method called GetMirrorURL, which takes no parameters. If a service chooses not to provide mirroring (perhaps it uses a proprietary database or algorithm), it should return null in response to such a query. Services that wish to provide mirroring will return the mirror URL. Once the mirror URL is received by the client, the client then makes an HTTP/1.1 request for that URL. Because each client may support one or more different formats for service executables (i.e. Java, PHP, .NET, etc.), the client must include an `Accept` header with the request, specifying a media range of those MIME types which the client is willing to accept. The server will have a parallel set of possible formats that the service to be mirrored can be provided in (and may support automatic translation between formats). If a mutually agreeable format can be found, the server will return the service executable in that format, along with a `Content-Type` header giving the MIME type of the format chosen. The service must guarantee that all formats return identical output for identical input.

The server should also include an `Expires` response header which gives the time at which the service content expires. It is the responsibility of the client to re-query the service at the time of first invocation after the date given in the `Expires` response header to see if a new version is available. The client should use the `If-Modified-Since` request header to prevent re-downloading the service if it is unchanged. The decision support service provider may need to expire the content before the `Expires` header date is reached (in the case, for example, of a drug recall), and a mechanism is provided for this. After successfully downloading the mirrored service executable, the client should register with the service by invoking the `RegisterMirror` function, and providing a URL on a server controlled by the client but reachable by the decision support service. If the service provider wishes to expire the content early, they would POST a blank message to the URL provided. The POST method was chosen because RFC 2616 specifies that the GET method should not have side effects (i.e. that no change in state should occur as a result of calling the GET method). Registering for expiration notifications is not a replacement for honoring the `Expires` response header, because receiving the expiration notification is not guaranteed – if the service provider cannot successfully complete the expiration notification on its first attempt, it is not required to continue attempting the notification.

In most cases, mirroring should be transparent to the client. This would normally be accomplished by a proxy. With a proxy, all requests for local, remote or mirrored services are sent through the proxy. When the proxy detects frequent queries to a remote service, it automatically attempts to mirror that service locally. If successful, future queries to that service are rewritten so they call the local mirror without needing to update the client.

While the technical details of the mirroring mechanism are complex, its purpose is simple: to provide clients the ability to run remote content locally. This is likely to alleviate four concerns that potential users may have about a service-oriented architecture: performance (local services run faster), accessibility (local services can be accessed even if the external internet connection is down), security (local services don't require that patient data be provided to outside parties) and concerns about external factors (local services won't become unavailable if the service provider goes out of business, or if there is a dispute between the client and the service provider).

In certain cases, providers of decision support services may not want to allow mirroring. For example, the service may use proprietary data or algorithms which the provider would not want to expose to clients. Or, in the case of fee-based services, the provider may not wish to allow mirroring because a mirror would allow the client to use the service even if they stopped paying for it. Because of these issues, mirroring is not a required capability, but it is expected that clients would prefer having the capability, and might preferentially choose services which offer it.

CHAPTER 5: A FRAMEWORK FOR EVALUATION

A project such as this one, with both theoretical and applied contributions, requires a tailored evaluation framework. In order to develop such a framework, I worked closely with my committee. We reviewed a number of representative dissertation in medical informatics and computer science with similar approaches to my own as well as two key works on evaluation in medical informatics and experimental computer science: a 1994 report from the National Academy of Sciences, *Academic Careers for Experimental Computer Scientists and Engineers* (158) and William Stead's "Designing medical informatics research and library-resource projects to increase what is learned" (159). Because of their centrality to the final evaluation framework that was developed they are briefly reviewed in this chapter.

5.1 ACADEMIC CAREERS FOR EXPERIMENTAL COMPUTER SCIENTISTS AND ENGINEERS

In 1994 the National Academy of Sciences released a report titled *Academic Careers for Experimental Computer Scientists and Engineers*. This report was written in response to a tension between theoretical and academic computer scientists about what sort of research constitutes "real science." While it primarily discusses career-related issues such as promotion and tenure, it also has a chapter on evaluation. It considers evaluation not from the perspective of methods, as is typical, but instead from the perspective of dissemination. While it acknowledges that papers in refereed academic journals represent the pinnacle of dissemination, it discusses other forms of dissemination which can also form valuable contributions to knowledge (158):

- Conference papers
- Demonstrations
- Sharing source for software
- Providing access to experimental computer systems
- Sharing chip designs
- Disseminating graphic images and CAD tools
- Providing access to data

This dissertation meets the evaluation standard set out by *Academic Careers for Computer Scientists* because the taxonomy paper has been accepted for publication in a major, refereed informatics journal and another submission describing the architecture and evaluation results is now under preparation. However, several of the alternative dissemination methods have also been employed. The architecture has been demonstrated at four academic medical centers (Cincinnati Children's Hospital, Columbia, the Regenstrief Institute and Partners), the source code is available under an open source license, and a manuscript describing the architecture has been submitted to the 2007 AMIA student paper competition.

5.2 THE STEAD FRAMEWORK

One of the earliest and most widely read papers on evaluation in medical informatics is "Designing medical informatics research and library-resource projects to increase what is learned" by William Stead and the members of the Biomedical Library and Informatics Review Committee (BLIRC). The BLIRC is the board that reviews grants submitted to the National Library of Medicine. The BLIRC developed a consensus framework to "increase what is learned from information access, information systems, and medical informatics research projects" by improving the quality of evaluation in the field of informatics. The framework is summarized in Table 7 which is reproduced from the paper.

		Evaluation				
		I	II	III	IV	V
			Laboratory		Remote Field	
	System Development	Definition	Bench	Field	Validity	Efficacy
A	Specification	→	↓			
B	Component development		↓			
C	Combination of components into a system		→/↓	→/↓	↓	
D	Integration of system into environment		→	→/↓	→/↓	↓
E	Routine use			→	→	→

TABLE 7 - STEAD'S EVALUATION FRAMEWORK (REPRODUCED FROM STEAD, ET AL. (159)).

This table sets out five levels of project development (from specification through routine use) and five levels of evaluation (from definition to remote field efficacy). At the vertices, the researcher is directed to move further in system development, move further in evaluation, or take either option. The goal of the evaluation framework is to ensure that the evaluation level is appropriate to the level of system development.

5.3 THE FRAMEWORK

Based on this background review, my committee and I developed a five-phase evaluation framework:

1. **Feature determination:** Determine a set of desirable features for a decision support architecture, and describe the relative ability of the SANDS architecture and other architectures to exhibit these desirable features.
2. **Existence:** Develop an architecture for decision support which exhibits this set of features, and demonstrate that the architecture is feasible by building a proof-of-concept prototype of it.
3. **Utility:** Demonstrate that the the architecture is useful by showing that it can be integrated with existing decision support systems.
4. **Coverage:** Demonstrate that the architecture is sufficiently general by measuring how well it covers a large knowledge base of decision support content and compare its coverage to the coverage possible with other architectures such as Arden Syntax, GLIF, SEBASTIAN and SAGE.
5. **Performance:** Demonstrate that the architecture is fast enough to enable a reasonable response time in applications.

Based on Stead's framework for development, the taxonomy and specification form a level A (specification) project, while the proof-of-concept is at levels B (component development) and C (combination of components into a system).

The first and second phase of evaluation, **feature determination** and **existence**, cross phases I (definition) and II (laboratory bench) of evaluation, and begins to lay the groundwork for further review. The third phase, **utility**, crosses laboratory bench and field, and ventures into remote field validity because the third-party services that were integrated were outside the control of the research team, which is the defining point of remote field evaluation. Phases 4-5 of the evaluation plan, **coverage** and **performance**, actually track the same evaluation levels as phase 3, but provide additional dimensions to measure along.

Taken together, the five phases of evaluation satisfy both the Stead and NAS frameworks while also providing a way to think about the feasibility and usefulness of the SANDS architecture and its relative advantages and disadvantages when compared to other approaches for sharing decision support content.

CHAPTER 6: RESULTS

6.1 FEATURE DETERMINATION

The driving force behind developing a new architecture for clinical decision support was a recognition that other existing architectures had limitations which impaired their ability to be maximally effective. Based on a review of the history of decision support systems and architectures (discussed in Chapter 1 of this dissertation), as well as a review of some best practices for decision support, a set of desirable features for a decision support architecture was identified. These are given in Table 8, and described in detail below. One fundamental question is whether any of the major architectures for decision support could satisfy all of these features. All four the architectural approaches were considered in the review: standalone, integrated, standardized and service-oriented. For standalone and integrated architectures, hypothetical best-in-class were considered – if it is possible for any system developed in that architectural paradigm to exhibit a feature, that feature is credited to the hypothetical system for that paradigm, whether it actually currently exists or not. For standards-based systems, Arden Syntax and GLIF 3.5 were used. GLIF offers a choice of expression languages, with GELLO being the recommended choice. We used GELLO for three reasons: first, because it's recommended, second, because it's the most expressive expression language for GLIF, and third, because there is some extremely promising work on developing a GELLO toolkit and authoring environment underway (121). For service-oriented systems, SEBASTIAN and SAGE were evaluated. The results of the comparison are presented in Table 8.

Feature	Stand-alone	Integrated	Arden	GLIF 3.5 / GELLO	SEBASTIAN	SAGE
Avoids vocabulary issues	x	x				x
Shareable	x		x	x	x	x
Can view decision support content	x	x	x	x		x
Content separate from code			x	x	x	x
Central updates					x	
Explicitly designed for multiple patient data sources						
Explicitly designed for multiple CDS sources					x	
Content integrated into workflows		x	x	x	x	x
Supports event driven CDS		x	x	x	x	x
Supports non-event driven CDS	x	x				
Support decision support across multiple patients	x	x				x
Enables separation of responsibilities	x		x	x	x	x
Enables composition of services		x			x	x
Allows black-box services	x				x	x
Free choice of programming language	x				x	x

TABLE 8 - FEATURE COMPARISON FOR A VAREITY OF DECISION SUPPORT ARCHITECTURES.

The dimensions of comparison are:

- **Avoids vocabulary issues:** One key challenge of decision support is vocabulary issues. Standards like Arden Syntax create vocabulary issues by leaving vocabulary undefined. Stand-alone systems, however, avoid them entirely by not interfacing with other systems. SANDS avoids such issues by tying its vocabulary requirements to those standards approved by HITSP, so there is no additional effort required of clinical system developers to support the vocabulary needs of SANDS, so long as they support these standards (such support is beginning to be mandated by the (voluntary) CCHIT certification process).

- **Shareable:** The basic goal of most knowledge representation formalisms is enabling the sharing of decision support content. Each approach, except for fully integrated systems, enables this. Standalone systems are trivially shareable, since they can simply be copied from one system to another.
- **Can view decision support content:** Formalisms like Arden Syntax and GLIF specify a format for sharing knowledge that is both human and machine readable. This is generally a good thing because it allows consumers of content to review it. However, in certain cases, a content developer may not want to allow consumers to view the content (e.g. if they consider their content proprietary). While the developer of a SANDS service always has the option to show the code to the consumer, this ability is not guaranteed in SANDS.
- **Content separate from code:** A common goal of knowledge management and software engineering is the separation of content (decision support rules) from the code for the clinical system. All of the architectures, except the fully integrated model, make this separation explicit. In the case of the integrated model knowledge and code are sometimes mixed together.
- **Central updates:** One challenge in clinical decision support is keeping decision support systems up to date, particularly since medical knowledge changes quickly. With SEBASTIAN and SANDS clinical decision support content is held with its developer, allowing it to be updated centrally.

- **Explicitly design for multiple patient data sources:** Patient data is commonly spread across multiple clinical systems – for example, a patient may have data in their primary care provider's system, a specialist's system and a hospital system from a hospital admission. SANDS is the only approach which is explicitly designed to handle multiple patient data sources which efforts such as RHIO's and the NHIN are intended to address.
- **Explicitly designed for multiple CDS sources:** SANDS and SEBASTIAN are both designed to allow for multiple decision support services, and both support mechanisms for choreographing and integrating these services. This is important, particularly when control of services is decentralized and distributed. For example, in the case of medication ordering, a clinical system might want decision support relating to drug interactions (which could be provided by a commercial knowledge vendor), formulary coverage (likely provided by the patient's insurer) and black-box warnings (which could be provided by the FDA).
- **Content integrated into workflows:** In the ideal case, decision support is closely integrated into clinical workflows. All of the approaches for clinical decision support, with the exception of standalone systems, provide this capability.
- **Supports event driven CDS:** Most clinical decision support is event-driven – some event in a clinical system (such as ordering a medication or receiving a lab result) triggers the decision support to fire. All of the approaches support event-driven decision support, except for standalone systems, which the user must explicitly invoke.

- **Supports non-event driven CDS:** Certain cases of clinical decision support are not necessarily event driven – for example, a system which brings up a list of all diabetic patients who haven't had a hemoglobin A1C test in the past year could be a form of decision support, even though it lacks a specific triggering clinical event. While this use case can be easily handled in standalone and integrated systems, the other approaches have varying levels of support. Arden Syntax can only handle event-driven decision support. GLIF has reasonable support for non-event driven logic, and SAGE and SEBASTIAN can be made to perform non-event driven logic. SANDS explicitly allows for it.
- **Support decision support across multiple patients:** Most decision support (and most decision support systems) is designed around a single patient at a time. However, useful systems can be built which reason across multiple patients, such as the hemoglobin A1c panel viewer described previously, or the public health use case reviewed in Chapter 4. Standalone and integrated systems enable this but Arden, GLIF and SEBASTIAN do not. Support for such systems is explicitly designed into SANDS and SAGE.
- **Enables separation of responsibilities:** The people who develop a clinical system (often computer programmers) and the people who develop decision support content (usually clinicians and informaticists) are frequently different. As such, it is helpful if these responsibilities can be separated. This is possible in any of the approaches, but it becomes challenging when decision support content is hardcoded directly into a clinical system, as can be the case in the integrated approach.

- **Enables composition of services:** It is sometimes the case that one decision support system might want to invoke another decision support system. This is explicitly considered in the three service-oriented approaches (SEBASTIAN, SAGE and SANDS) and is also possible with integrated systems. However, Arden Syntax does not enable this capability very directly, and it can be difficult to achieve with standalone systems unless their inputs and outputs are synchronized.
- **Allows black-box services:** In certain cases decision support developers may consider the logic of their systems to be proprietary. In the case of integrated systems, Arden Syntax and GLIF, the developers must expose their logic to the consumers of their system. However, standalone systems and the three service approaches allow for black-box services, where the logic is not exposed. This brings the large caveat that, in a black box service, the consumer cannot review or validate the logic, so choosing to consume such a service must be carefully reasoned, but is, perhaps, justifiable if the service comes from a highly reliable source.
- **Free choice of programming language:** Integrated systems, Arden Syntax and GLIF all require decision support developers to use a specific language for representing their content. In many cases this imposes little limitation, but if the decision support developer is not comfortable working in that language, or if the language does not support the type of logic that the developer intends to use, this can be a very high cost. This is not faced in standalone systems, since they aren't designed to integrate with clinical systems and it is also avoided by the three service architectures, since they impose only interface requirements.

Overall, this analysis indicates that each of the existing architectures has gaps when compared against this list of desirable features. While some of these gaps could be resolved simply by adding features to the existing architectures, others are structural – for example, a fully integrated system, by definition, cannot use a central knowledge service. The SANDS architecture is intentionally designed to support all of these desired features.

6.2 EXISTENCE

Now that the SANDS architecture has been described and shown to support a set of useful features, the next logical question is whether is feasible. The goal of the existence phase is to demonstrate the feasibility of the SANDS architecture by building a working prototype of it – an existence proof. The prototype is described in detail in Chapter 3, and the protocols and messages used are described in detail in Appendices A an B. Further, open source code is available so that others can run and use the architecture. The code can be found at http://medir.ohsu.edu/~wrightad/sands/. Demonstrations of the architecture have been conducted at a variety of sites around the country.

6.3 UTILITY

The next step beyond existence is utility. The working prototype shows that the architecture is feasible, but can it do useful things? In this section, two dimensions of utility are considered: clinical utility (can clinically useful interventions be developed in a system like SANDS) and functional utility (does SANDS allow for a useful variety of different interventions).

In this dissertation, clinical utility is measured based on inherited utility from the use cases – most of the use cases comprise decision support interventions whose utility has been demonstrated in previous studies. To the extent that SANDS provides the ability to implement these proven interventions, it inherits their clinical utility. We will review the clinical utility use case by use case:

- **Drug interaction checking:** Significant evidence exists that medication ordering errors (including drug-drug interactions) are extremely common, and that these errors are associated with morbidity and mortality (127-130). There is also evidence that decision support, such as drug interaction checking, can reduce the frequency of both prescribing errors and resultant adverse events (51, 104, 131, 133, 134, 136, 138, 139). SANDS provides a new way to tightly and conveniently integrate drug interaction checking into the ordering process.
- **Syndromic surveillance:** Syndromic surveillance is a high priority and automated electronic systems for detecting disease outbreaks can be very effective (160-167). SANDS enables reporting of clinical diagnoses from electronic health records (rather than administrative codes, which are more commonly reported), and also allows specialized public health information systems to tightly integrate with the same network.

- **Diagnostic decision support:** Since the beginning of the field of informatics, researchers have developed diagnostic decision support systems. The available evidence suggests that, while such systems are frequently impractical or unwieldy, they are generally as accurate as, or more accurate than human clinicians, and they can help clinicians improve their diagnostic accuracy (10-18, 23, 24, 29, 32-35, 39, 45, 53, 58, 141-144, 168-174). In this particular use case SANDS has been used to integrate the Isabel Diagnostic Reminder System (Isabel Healthcare, Reston, Virginia) with an electronic health record, and further evidence exists that Isabel is clinically useful (141, 142, 144) when so integrated.
- **Inappropriate prescribing in older adults:** Inappropriate prescribing in older adults is known to be common (146) in spite of available guidance about avoiding such prescribing (145). In this dissertation a service was created to enable automated checking of prescriptions against this guidance.
- **Information at the point of care:** It is well known that clinicians have many questions at the point of care (149, 151) and good information tools exist to help them resolve these questions (154-156, 175-179). SANDS makes it easier to tightly integrate these tools into the clinical workflow.
- **Personal health record:** Personal health records have great promise for improving patients' understanding of their health, their sense of self efficacy and their compliance with their healthcare plans (180-184). The personal health record use case in SANDS shows the feasibility of creating a composite personal health record

based on an NHIN, and providing patient-facing clinical decision support through that personal health record.

In addition to clinical utility, SANDS provides functional utility: the architecture allows for use cases with a variety of functional properties, as shown in Table 9.

	Diagnostic Support	POC Info: Google	POC Info: UpToDate	Inappropriate Prescribing	Drug-drug interaction	Syndromic surveillance	Personal Health Record
Developer	Isabel	Google	UpToDate	Self	Lexi-Comp	Self	Self
User	Clinician	Clinician	Clinician	Clinician	Clinician	Clinician, epidemiologist	Patient
Information source	Textbooks	Internet (expert curated)	Expert writers	Expert panel	Lexi-Comp pharmacists	State Health Dept.	Other data sources
Clinical purpose	Diagnosis	Diagnosis / therapy	Diagnosis / therapy	Drug therapy	Drug therapy	Public health	Patient information
Inference type	NLP	IR	IR	Rule-based	Rule-based	Rule-based, geospatial	Compositional
Composition role	None	Provider	Provider	Provider	Provider	Consumer	Consumer
Business model	Subscription	Ad-supported	Subscription	Public service	Subscription	Government supported	Unclear: public service / subscription / third party
Pay	Yes	No	Yes	No			
Status	Already a service	Already a service	Wrapper	De novo			

			pay
Pay	Yes	No	No
Status	Already a service	De novo	De novo

TABLE 9 - PROPERTIES OF SANDS PROTOTYPE USE CASES.

The properties in Table 9 are:

- **Developer:** Who developed the decision support system. Table 9 shows that use cases from a variety of commercial developers, as well as self-developed use cases have been prototyped in SANDS.
- **User:** The intended user of a decision support system. Most decision support is targeted at clinicians, but use cases targeted at public health departments and patients have also been prototyped.
- **Information source:** A variety of information sources are demonstrated, ranging from staff experts in a number of the commercial cases, to the internet or government agencies in other use cases.
- **Clinical purpose:** Use cases crossing diagnosis, therapy, information display and public health have all been implemented in the SANDS prototype.
- **Inference type:** A variety of inference types are demonstrated, ranging from simple rule-based systems to complex NLP interventions. This is particularly significant because many current knowledge representation formalisms are limited to rule-based interventions. Since SANDS imposes no restrictions on the way decision support interventions are implemented (so long as they implement the SANDS interface), it is possible to develop a service using any arbitrary inference type in SANDS.

- **Composition role:** One key advantage of SANDS is that services can be composed of other services. This is demonstrated in a number of ways in SANDS. For example, the public health service calls a geo-coding service, while the PHR use case calls a number of other services, including the point-of-care information tools and the drug interaction checker. These services are then used to provide patient-tailored information on medications and drug interactions.
- **Business model:** A variety of business models are demonstrated, including commercial services, services which might be provided by the government and services which would be provided by medical specialty societies (who currently provide written guidelines).
- **Pay:** Both pay and free services have been prototyped.
- **Status:** Some of the use cases were already available as a service (which could be easily translated to the SANDS protocols), while others required development of a service wrapper and some were developed *de novo*.

As shown in Table 9 and the accompanying discussion a wide variety of clinical decision support types can be implemented in SANDS. This is quantified in section 6.5. Combined with the clinical utility of the decision support interventions described previously, this analysis suggests that SANDS has high utility – it can be employed to carry out useful tasks.

6.4 COVERAGE

The next element of the evaluation framework is coverage – the ability of the SANDS architecture to encode clinical knowledge in comparison to other approaches. The basis for the coverage metric is the large knowledge base of clinical decision support content described in Chapter 3. This knowledge base spans several hospitals and a large number of outpatient clinics, and contains 181 rule types and 7,120 rule instances. SANDS is compared against a variety of architectures for sharing decision support content from all four of the architecture phases described in Chapter 1. For the first two phases – standalone and integrated, a hypothetical best-in-class system is considered. For stand-alone, this means a system which can use any kind of data element, but with the significant limitation that decision support can only be triggered by user request. The hypothetical integrated system is actually the gold standard – when decision support is hard-coded directly into a clinical system there is no limit to what can be done (although there are a variety of disadvantages to fully integrated systems, as described in Chapter 1 and the next section of this chapter). For standards-based integrated systems, Arden Syntax and GLIF are used as representatives, since they are the two most widely used and studied approaches. Both of the existing service-oriented approaches, SEBASTIAN and SAGE are also included in the comparison. Table 10 shows the capability mapping between the taxonomy and the comparison systems, as well as SANDS.

	Stand - alone	Integ- rated	Arde n	GLIF	SEBAS -TIAN	SAGE	SAND S
Trigger							
Order entered		x	x	x	x	x	x
Lab result stored		x	x	x	x	x	x
Outpatient encounter		x	x	x	x	x	x
User request	x	x		x	x	x	x
Time		x				x	x
Admission		x	x	x	x	x	x
Problem entered		x	x	x	x	x	x
Enter allergies		x	x	x	x	x	x
Enter weight		x	x	x	x	x	x
Data element							
Lab result / observation	x	x	x	x	x	x	x
Drug list	x	x	x	x	x	x	x
Hospital Unit	x	x	x	x	x	x	x
Diagnosis / Problem	x	x	x	x	x	x	x
Age	x	x	x	x	x	x	x
Non-drug orders	x	x	x	x	x	x	x
Gender	x	x	x	x	x	x	x
Family history	x	x	x	x	x	x	x
Allergy List	x	x	x	x	x	x	x
Weight	x	x	x	x	x	x	x
Surgical history	x	x	x	x	x	x	x
Reason for admission	x	x	x	x	x	x	x
Prior visit types	x	x	x	x	x	x	x
Race	x	x	x	x	x	x	x
Intervention							
Notify	x	x	x	x	x	x	x
Log		x					x
Provide defaults / picklists		x				x	x
Show Guidelines	x	x		x	x	x	x
Collect freetext		x				x	x
Get approval		x					x
Show data entry template		x				x	x

	Stand - alone	Integ- rated	Arde n	GLIF	SEBAS -TIAN	SAGE	SAND S
Offered choice							
Write order		x		x	x	x	x
Defer warning		x		x		x	x
Override rule / keep order		x		x	x	x	x
Cancel existing order		x		x	x	x	x
Cancel current order		x		x	x	x	x
Edit current order		x		x	x	x	x
Edit existing order		x		x	x	x	x
Set allergies		x		x	x	x	x
Write letter		x				x	x
Write note		x					x
Edit problem list		x		x	x	x	x
Enter weight, height or age		x		x	x	x	x

TABLE 10 - FUNCTIONAL CAPABILITIES OF VARIOUS DECISION SUPPORT ARCHITECTURES.

The mapping in Table 10 was then applied to the entire database of decision support content to compute metrics, seen in Figure 25. These coverage metrics represent the percentage of rules in the database which can be represented in the given architecture. For any given rule to be representable in an architecture, each of the functions that the rule requires (in terms of triggers, data elements, interventions and choices offered) must be provided by that architecture.

As Figure 25 indicates, SANDS achieved coverage equal to the gold standard – fully integrated systems. This, of course, is by design – SANDS was intentionally developed to have full coverage of this knowledge base. SAGE also performed very well. GLIF and SEBASTIAN were able to map slightly less than half the content in the database. Their performances were very similar largely because they employ similar data models. Arden Syntax had much lower coverage, mapping only slightly more than 15% of the database. This is largely attributable to the fact that Arden Syntax

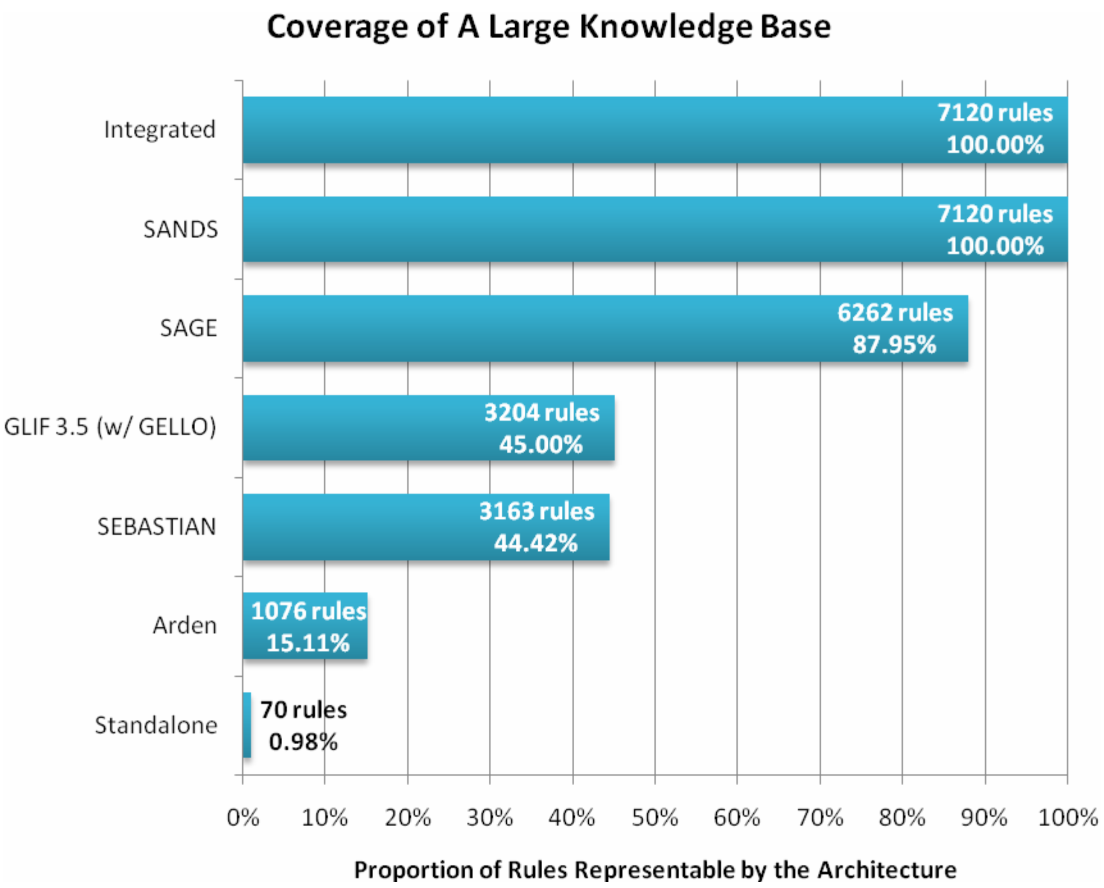


FIGURE 25 - COVERAGE METRICS FOR VARIOUS DECISION SUPPORT ARCHITECTURES.

supports only simple notifications – it does not allow for offering actionable choices to the user in response to a notification. If this limitation is excluded from the analysis, Arden Syntax's coverage score increases to 45.88%. Standalone systems, as expected, offer the poorest coverage because they don't support triggers or actionable choices, so only very rudimentary decision support rules (those without a triggering event, and with no response actions) can be fully represented.

It is important to note that this coverage analysis should not be viewed as a competition between SANDS and the other approaches. Each of the approaches (with the exception of standalone systems) could be extended to achieve 100% coverage, and it is likely that some of them will be as a result of this analysis. Instead, the purpose of this evaluation is two-fold: first, to show that the SANDS architecture is sufficiently general to map a wide variety of decision support content types, and second to show that there are, at present, remediable gaps in the other architectures. Further, one important intent of SANDS is that it does not restrict the developer's approach to knowledge representation and execution, so SANDS is not intended to supplant other knowledge representation formalisms. Instead, it is entirely possible (and has been roughly prototyped) to develop decision support systems for SANDS in any of these formalisms. Such an approach would be a natural way to extend any of the other formalisms to 100% coverage. For example, if the medical and action ontologies in GLIF were replaced by the SANDS taxonomy (and GLIF is designed with enough generality to allow for such a substitution), this would close the coverage gap.

6.5 PERFORMANCE

One critical factor of any decision support system is performance. A common goal of clinical system developers is sub-second response time (185) and one critical question we faced was whether this ideal could be achieved using the SANDS architecture. The SANDS architecture is subject to five kinds of delay which are generally additive:

- **Network latency:** The time it takes for a packet to propagate between two hosts on a network. This is a startup cost of transmission – after the first packet, throughput is governed by transmission delay.
- **Transmission delay:** The time needed to transmit a message over the network once latency is overcome. With SANDS' small message sizes, throughput is usually not a major source of delay.
- **Patient data fetch:** SANDS fetches a patient's clinical data from the NHIN. The cost of this fetch can be fairly high so SANDS employs a caching strategy to reduce this delay.
- **Parsing overhead:** SANDS is based on XML protocols and there is overhead in parsing the XML. Given the size of the messages used in SANDS, the overhead is fairly small.
- **Inference time:** This is the actual time that the inference takes to run. This is not overhead added by SANDS. Even tightly integrated decision support systems face this delay.

The latency delay of a typical transmission varies significantly depending on the distance between the client and server. Within a subnet, it is generally less than one millisecond. Within the continental United States, I observed times ranging from 3ms to 170ms, with typical transmissions seeing about 75ms latency delay. Because the amount of data SANDS transmits is small, transmission delay tends to be very small, on the order of 1-3ms. The patient data fetch time is a function of the NHIN architecture and, as discussed in Section 4.3, tended to be unacceptably slow (on the order of 6-10 seconds). As a result, caching was employed, which reduced the fetch delay to zero (when stored locally in the service), or on the order of 100-200ms, when stored in the cross-service distributed cache described in the same section. The parsing overhead for SANDS messages was approximately 10ms using Microsoft's native XML parser. The first four delays are all overhead added by SANDS. The final source of delay, inference time, is a property of the decision support service, and was the most variable. Simple services could complete their inferences very quickly (from 50-100ms), which other services took nearly a second.

To assess the robustness of the SANDS architecture and the delays associated with it we conducted a timing and reliability study. We set up the system to automatically poll each service in the SANDS implementation every 5 minutes (24 hours/day, 7 days/week) over a continuous four week period (a total of 13,440 requests). Table 11 shows the results of this study. The "Avg Time" column gives the mean response time in milliseconds for each service. The next four columns give the frequency with which various results occurred: if the query failed, if it returned within 1 second, if it returned within 5 seconds, or if it was successful but took 5 or more seconds to complete. Seven of the nine (78%) SANDS decision support modules had subsecond response times > 97% of the time. The prescribing in the elderly service was unavailable for two brief periods because it was being reprogrammed – these periods are excluded from the analysis.

		Avg Time	Result			
Service	**n**	**(ms)**	**Failed**	**<1s**	**<5s**	**>=5s**
Self-Developed Services						
Prescribing in the elderly	21,711	73.32	0.00%	100.00%	100.00%	0.00%
Syndromic surveillance	24,765	73.58	0.00%	100.00%	100.00%	0.00%
Externally-Developed Services						
Diagnosis (Isabel)	24,765	524.61	0.03%	99.43%	99.95%	0.02%
Information (Google)	24,765	192.45	0.03%	99.23%	99.89%	0.08%
Information (UpToDate)	24,765	788.40	0.38%	94.61%	99.07%	0.55%
Drug-drug interaction	24,765	1382.17	0.02%	0.00%	99.62%	0.36%
Google geocoder	24,765	137.90	0.06%	99.72%	99.94%	0.01%
Yahoo geocoder	24,765	250.57	0.14%	98.07%	99.71%	0.15%
Geocoder.us	24,765	262.76	0.63%	98.51%	99.23%	0.14%

TABLE 11 - PERFORMANCE OF SANDS PROTOYPE SERVICES.

Overall, these results indicate that it is not only possible, but highly reliable and quick, to perform a variety of useful clinical decision support interventions using the SANDS architecture with sub-second response time as the standard. Most (7/9) services consistently (>97% of the time) serviced requests in under one second. The syndromic surveillance and prescribing in the elderly services, which were self developed, had outstanding performance because they were optimized for efficiency and located locally so a request to them did not have to traverse the Internet. The drug-drug interaction service never completed within 1 second, but almost always (99.83%) completed within 5 seconds. This is likely due to the data structures used by the developer of the service.

The last three services, called Geocoders, merit further comment. The syndromic surveillance use case has a mapping capability, where cases of reportable diseases are displayed geospatially. To do this, a service called a Geocoder is used. A Geocoder takes an address and finds the corresponding latitude and longitude. Three different free Geocoding services were tested and the variation in performance was striking – Google's Geocoder almost never failed, and returned within 1 second 99.9% of the time. Yahoo's Geocoder, doing the same task, failed twice as often and took, on average, nearly twice as long, as seen in Table 11. The implications of the variable performance of these three identically purposed services are more fully discussed in section 7.7. Clearly, system reliability and performance will be an important consideration for any user of the SANDS architecture. While the architecture itself imposes only minimal latency and overhead, the performance of the same task, implemented by different developers, may vary greatly so it will be important to carefully evaluate performance before choosing services.

6.6 SUMMARY OF RESULTS

As discussed in the preceding sections, SANDS performs favorably on a variety of metrics. I demonstrated that the architecture is feasible, that it has a variety of desirable features sometimes lacking in other architectures for decision support, that it has excellent clinical and functional utility, that it is able to successfully model a large knowledge base of decision support content and that its performance is more than adequate for sub-second response time.

CHAPTER 7: DISCUSSION

7.1 CHALLENGES IN INTEGRATING DATA

As described in section 4.2, I developed a Patient Data Class Library that integrates data from the NHIN and prepares it for easy use in decision support. The data elements represented by the Patient Data Class Library were:

Demographics and vital signs:

- First Name
- Last name
- Gender
- Race
- Date of birth
- Weight
- Height
- Care setting

Clinical information:

- Observations
- Drugs
- Problems
- Family history
- Procedure history
- Allergies
- Other orders

The standards used and challenges in representation were discussed in section 4.2. However there were also some difficulties in integrating patient data. For demographics, discrepancies were uncommon and usually of little significance (for example, discrepant spellings of the patient's first name). When present, they were resolved by using the most recent observation as the correct one. Vital signs were resolved in a similar fashion: all vital signs were stored and made available with timestamps as observations. For height, weight and care setting the most recent value was also made available as a discrete value in the demographics component of the Patient Data Class Library

Integrating the clinical information was more difficult. For some data elements, such as observations and procedure history, the correct integrated set is simply the union of all non-duplicate elements stored in clinical systems on the national health information network.

However, for drugs in particular, the problem was much more difficult. We often encountered patients who had medication lists spread across a variety of clinical systems, often containing the same drug with different doses or two drugs in a therapeutic class where co-administration is very uncommon. We were not able to accurately reconcile such medication lists in a fully automated fashion. This problem is well understood and described in the literature (186, 187) and is the focus of a number of active research projects, such as the RxSafe project in Lincoln City, OR (188). It is likely that any comprehensive solution will require human intervention, whether it's a clinician to reconcile the medication lists, a pharmacist or the patient himself.

That said, for many clinical decision support applications a fully reconciled medication list is unnecessary. Consider, for example, drug interaction checking. Assume that the patient has a three-year-old order for warfarin entered by another provider with no updates in any other systems. While it may be very probable the patient is no longer on warfarin; if a current provider wrote an order for, say aspirin, it would still be sensible to show an alert that the contemplated order poses a serious contraindication with the old warfarin order. The provider could then query the patient to determine if the patient was still taking warfarin and act accordingly.

For certain types of decision support, however, this may not be adequate. Consider, for example, a hypothetical pill burden calculator. Unless the calculator has an accurate way of determining what medications a patient is still taking calculating a correct pill burden would be very difficult. Luckily this problem cuts across all uses and users of integrated medication data, so it is likely to see significant attention in the future.

7.2 DRUG TERMINOLOGY CHALLENGES

Many systems, including the national health information network prototype that I used in this dissertation use national drug codes or NDCs. These codes are popular because they are the basis of most pharmaceutical claims. Since nearly all pharmaceutical claims are processed electronically they tend to be much more widely available than prescription data because electronic prescribing is still rare. The most significant difficulty with NDCs are their specificity. Each individual dose form and quantity of a medication is assigned a unique national drug code. That means that a common generic drug, like aspirin, may have dozens or hundreds of applicable NDCs. This creates a challenge for decision support, because any rule that references a given drug by its NDC must be aware of the fact that there could actually be many NDCs for that drug, and that more may appear in the future. A further challenge is that NDCs can be unreliable. The code space for NDCs is managed in a decentralized fashion, with the FDA assigning parent codes to all manufacturers of pharmaceuticals. These manufacturers then assign NDCs from their code space whenever they produce a new drug. Unfortunately, there are cases where manufacturers recycle drug codes, or simply assign the same drug code to multiple drugs. And while manufacturers are required to report their assignments back to the FDA, there are cases where they fail to do so. As such, even in our limited prototype we encountered a number of situations where NDCs found on a patient's drug list could not be matched by decision support systems. For example, the NDC code 00093086301 represents one form of ciprofloxacin. When this code was passed to the Lexi-Comp drug interaction service the query failed. Although the service, of course, contains data on ciprofloxacin, this particular NDC for Ciprofloxacin was not in its database. The same issue was encountered for NDC 68249020010 (for the drug Humibid, a brand name version of guaifenesin with potassium guaiacolsulfonate) and for NDC 00591049950 (a generic 100 MG doxycycline hyclate tablets produced by Watson Laboratories).

It is my strong recommendation that terminologies besides NDCs be used to describe drugs. The National Drug File Reference Terminology (NDF-RT) would be one good choice, but there are actually many. This cause is helped by the RxNorm project of the National Library of Medicine (90). RXNorm is designed to be a comprehensive map amongst various drug terminologies. While it includes NDCs, it faces the same limitations that other terminology systems based on NDCs have encountered: namely, that certain NDCs which have been assigned to drugs do not appear in RxNorm. Another ray of hope comes from the NCPDP SCRIPT electronic prescribing standard (87). Electronic prescriptions using NCPDP SCRIPT use robust drug vocabularies, and as more electronic prescriptions are written, the need to use claims data (where NDCs are most commonly found) is lessened.

7.3 PROBLEM LIST TERMINOLOGY CHALLENGES

While the largest terminological issue I encountered in the development of this prototype was related to drugs, problem list terminology also provided some interesting challenges. Perhaps the most widely used system for encoding problems is ICD-9. The ICD system, originally called the International Classification of Causes of Death was initially designed for use on death certificates. It has since been modified for clinical use, particularly in billing (the modified version is called ICD-9-CM, for clinical modification). Medical bills in the United States are generally comprised of ICD-9 codes, which describe the disease or indication for a procedure, and a CPT code which describes what the provider did. Many electronic health record systems have also adopted ICD-9 codes for use in problem lists. While this works in many cases, and certainly simplifies billing, it is frequently the case that the level of detail required for describing a problem for clinical purposes differs from the level of detail required for describing the same problem for billing purposes.

An initial version of the prototype system used ICD-9 codes because of their wide use and availability, and because the NHIN prototype used generally provides problem information in ICD-9 format. However, ongoing use of the prototype quickly showed that ICD-9 codes were inadequate for describing many common clinical problems, a problem which has been previously described in the literature (189). As such, I converted the problem list format for my prototype to SNOMED. Because SNOMED is a very large terminology designed for a variety of clinical purposes. I employed a subset of the SNOMED terminology designed for problem lists and developed by Kaiser Permanente and the Department of Veterans Affairs. This subset is frequently used for problem lists and is available either directly from the Veterans Affairs administration or via the UMLS. In addition to more clinically relevant problems, using SNOMED provides several significant advantages for decision support developers. SNOMED provides a rich semantic structure and hierarchy of concepts, so it's possible to develop rules that correctly handle generality. For example, SNOMED defines a heart disease concept so it's possible to develop a rule that says "patients with a history of heart disease should receive an influenza vaccination." Then, even if the patient's problem list does not contain the concept "heart disease", but contains a sub-concept such as "dilated cardiomyopathy" a rule engine that exploits SNOMED's concept network can correctly reason that the influenza vaccination rule still applies. While ICD-9 has its own concept hierarchy it is much less rich than SNOMED's and is designed for billing rather than clinical purposes.

7.4 MESSAGE AND SERVICE ARCHITECTURES

One of the early questions faced during the development of the SANDS architecture was whether to develop it as a message-based architecture or a service-oriented architecture. To understand the difference between the two architecture types it is perhaps easiest to consider an example. Consider an electronic prescribing transaction – a physician has decided to write a prescription for an antibiotic, so he or she enters the prescription into an EHR, and asks the EHR to transmit it electronically to the pharmacy. In a message-based architecture, the EHR would generate a message, formatted according to some agreed upon standard, and "release" it to a network, and that network would be responsible for properly routing the message and acting on it (probably with the help of some sort of message router or hub). In this model, the EHR gives no consideration to where the message goes (i.e. what pharmacy), or how it gets there. This relieves the EHR of significant responsibility, since it doesn't need to know what pharmacies exist, or how to reach them, but comes at the high cost of the EHR not being able to control how the message is distributed, or possibly even knowing if the message was delivered successfully or not. In a service-oriented model, the EHR would be much more proactive – it would connect to a prescribing service offered by a pharmacy (and would then, of course, need to know how to reach that pharmacy, or have a way to find out how to reach it), and would communicate directly with that service, providing the appropriate information needed, and having more fine-grained control of the service's behavior.

One common way of thinking about the difference between services and messages is that messages make data explicit but behavior implicit, while services make data implicit but behavior explicit. In healthcare, message-based architectures have predominated, particularly in the domains of reimbursement and intra-hospital clinical system integration. In the case of reimbursement, messages are generally routed by fee-based clearinghouses: RxHub and SureScripts for pharmacy claims using NCPDP message formats, and a variety of clearinghouses for medical claims, using HIPAA transaction message formats. Within a hospital, message architectures have also been predominant, usually based on HL7. All of the early HL7 standards were message standards – they defined a message format, and depended on a network to move the messages around. For this purpose, many hospitals have an internal message bus, such as the IBM Enterprise Service Bus, or Microsoft's BizTalk server, which is responsible for routing the HL7 messages. In addition to architectures based on a message bus, HL7 also supports point-to-point interfaces, which operate more similarly to services.

In IT, broadly, there has been a recent transition from mostly message-based architectures (which are viewed as an older technology) to service-oriented architectures, but this transition has been slower in HL7. A number of HL7 technical committees and working groups have been developing service specifications to complement messages, but this activity is not without controversy. In fact, although discussion within HL7 is usually fairly staid, there has been vigorous debate about the appropriateness and desirability of messages versus services. Much of this discussion has taken place on the HL7 SOA mailing list. The list is normally moderate-volume, with 407 threads over the past year. Most of those threads are devoted to technical details, but both the longest (with 46 posts) and second longest (with 20 posts) threads in the discussion were debates on the merits of services vs. messages. The next-runner-up had only 13 posts. It appears that resolution is not near for HL7, and that both service and message standards will be developed in parallel for the foreseeable future.

The issue of service-oriented architecture in medicine was also the subject of some recent debate in the *Journal of the American Medical Informatics Association*. In the March-April 2007 issue of the Journal, Kensaku Kawamoto and David Lobach, both of Duke, lay out an argument in favor of service-oriented architectures, while Prakash Nadkarni of Yale and Randolph Miller of Vanderbilt provide a counterpoint piece called "Service-oriented Architecture in Medical Software: Promises and Perils". Nadkarni and Miller acknowledge many of the benefits of SOA, but point out several pitfalls. First, they emphasize the importance and difficulty of developing standards, which has been a consistent focus of the SANDS effort. Every dimension of SANDS is carefully mapped to approved or emerging HITSP standards, and there are eventual plans to take SANDS itself into a standards process. Second, they raise the question of service discovery, which is an important one. As more and more services are made available (in any domain), cataloging and choosing amongst them can become challenging. For SANDS, I have chosen a fairly organic approach – developing an open architecture that allows anyone to contribute services to the catalog, and then relying on the judgment of consumers or expert curators to rank and choose amongst these services. In fact, SANDS explicitly allows for the development of services which exist simply to funnel queries to other services, which would allow, for example, a medical specialty society to provide an endorsement service, funneling queries to services the specialty society judged favorably. Other approaches, however, are possible, such as a centrally controlled system, where permission of some authority is required to provide a service. The third point that Nadkarni and Miller raise are the issues of reliability, liability and business models – these issues, as they relate to SANDS, are discussed in section 6.5 of this dissertation. The final point raised relates to control and commercialization. Kawamoto and Lobach have filed for a patent on their approach to service-oriented architectures for decision support, while simultaneously bringing it through a standards process. This tactic has some significant implications for anyone who might want to use this architecture. The approach I've taken with SANDS is simple – I will not seek any protection for the intellectual property I have developed and described here – I believe that any network for decision support should be free and open, and that no individual should control it. Anyone is free to develop SANDS services, or to integrate SANDS support into their clinical systems.

Ultimately, I'm not convinced that there is a single answer to the messages versus services debate. In many cases, equivalent functionality can be achieved with either model. However, for certain applications, like claims processing, messages seem more suitable, but for others, like decision support, services seem to offer better capabilities. In that light, it makes sense to take an open approach: choosing messages or services based on their appropriateness for a given task. For the SANDS architecture, I made the decision to use services rather than messages for three main reasons:

1. Messages require a clearinghouse for routing, and it is unclear who the clearinghouse for decision support would or should be. For claims, submitters have a clear incentive to pay a clearinghouse, since their claims won't be paid unless they do. The incentive for decision support is less clear. Also, whoever ran a decision support clearinghouse would have a great deal of power because they would control who could provide content over the network. A service oriented architecture is de-centralized – there is no broker, so anyone can offer content in the form of services, and consumers have the freedom to choose the service providers they want to access.
2. Most clinical decision support requires real-time, synchronous feedback: when a drug is ordered, that order needs to be transmitted to a decision support service, which will do its inference, and return a result, all within a short period of time. This is easy to achieve in a service-oriented architecture, which is based on the familiar function call paradigm. In a message-based architecture, it requires several messages traversing a network. In some cases message networks are designed to be best-effort-only, meaning that they don't guarantee delivery of a message, or they may employ queues, batching messages for later delivery. While this is

surmountable with special attention to the design of a message network, this paradigm is more naturally realized with a service-oriented architecture.

3. One of the most promising elements of a service-oriented architecture is the ability to compose services together to form new services (sometimes called service choreography). This is seen in the public health use case, where the decision support service calls a child service (the geocoder). One might imagine a variety of other services that could be created and choreographed – for example, an omnibus drug service which could query a variety of other services, like a drug-interaction service, an allergy service, a renal dosing service or a formulary checker, and then intelligently prioritize and assemble their responses. This sort of choreography is natural with a service-oriented architecture, but more difficult to implement in a message-based architecture.

7.5 ECONOMICS AND BUSINESS MODELS

At present, a number of restraints hinder the development of a full and robust market for clinical decision support. For most purchasers of clinical systems, such as electronic health records or computerized physician order entry systems, the easiest option for purchasing decision support content is to purchase it directly from their clinical system vendor. These vendors often have partnerships with preferred content providers. This creates a significant degree of vendor lock-in because if that vendor's customer wishes to employ a non-preferred clinical decision support system, they may be forced to develop costly interfaces. With open standards-based systems, such as

SANDS, purchasers of clinical information systems can instead adopt a building block model – purchasing the clinical information system of their choice and then choosing, application by application, the decision support systems that best meet their needs.

I anticipate that with an architecture such as this we would see a variety of business models for providing clinical decision support content. In some ways, one can conceptualize this architecture as being a sort of Internet for clinical decision support. Just as the Internet has a rich market of content suppliers ranging from paid providers of content to open source content, and even community edited content such as Wikipedia, so too could all of these models be supported in this decision support architecture. In many cases, we might expect that users would prefer to purchase decision support content from established vendors. However, it is also easy to imagine cases where individuals or organizations with an established interest in a topic might provide free decision support content. For example, many medical specialty societies have expressed interest in providing electronic forms of the paper guidelines they currently publish. An architecture such as this one would provide an open and high-leverage way for specialty societies to increase the adoption and use of such guidelines. One might further imagine other stakeholders that would be interested in providing clinical decision support systems. Consider for example the syndromic surveillance and disease reporting system described earlier in this dissertation. At present, public health departments spend a large amount of money developing disease tracking systems and publicizing them to encourage providers to report diagnoses of reportable diseases. Some of these resources could instead be dedicated to providing electronic decision support systems such as the one developed here which would automate portions of this process. Then, users of clinical systems who wish to automatically report reportable diseases could simply add this decision support service to their clinical system. Both the provider of a decision support system (in this case, the public health department) and the consumer of the service (in this case, a care provider) have strong incentives to cooperate around such a free decision support service.

Because of the exciting potential for a variety of business models to flourish I was careful to avoid being prescriptive about the business model or economic relationship between service providers and service consumers. Instead any provider or consumer who can communicate according to the decision support standard defined in this dissertation would be welcome to participate in an architecture such as this one. The business relationship, if any, between the provider and the consumer would be determined by the two parties.

One concern that arises when there is a business relationship between a provider of clinical information and a provider of clinical care is the issue of liability. In the 1980's, there was much debate in the field of informatics about whether clinical systems and clinical decision support systems should be regulated as medical devices, and whether there should be liability on the part of system providers for errors or omissions in their content or systems (174). While this question is still not settled law, the strong direction is that decision support content should be regulated and held liable using the same principles that apply to medical textbooks (which are, after all, a paper form of clinical decision support). In this paradigm, so long as there is a human intermediary between the device or system and the patient, that intermediary is the responsible provider, and there is no consequential liability for the system provider. That said, content providers and purchasers could certainly contract for additional indemnification – for example, a drug information provider might guarantee that their system will not miss any documented high severity drug interactions. This is a form of insurance, and it is possible that it, too, might become a basis of competition in a marketplace for clinical decision support. All things being equal, a decision support service which offered indemnity would be more attractive than one that didn't. If, on the other hand, there was an additional cost for this indemnity, providers and purchasers would have to conduct internal cost benefit analyses to determine the cost and value of such indemnity protection.

7.6 ORGANIZATIONAL ISSUES AND IMPLICATIONS

In addition to the economic implications, there are also organizational implications to consider with any clinical decision support or clinical system and this architecture is no different. The best evidence suggests that a phased approach to the implementation of any new clinical system or clinical decision support intervention is most likely to be successful (190). However, phased implementations are often very difficult to execute when using a monolithic system because the parts of the system often fail to function unless integrated into a whole. The SANDS architecture is inherently designed for extreme modularity. It would be conceivable that one might initially deploy a clinical system with no decision support turned on. After the issues surrounding the deployment settle out, the implementer could then slowly turn on clinical decision support modules as user experience and organizational priorities dictate.

Moreover, an architecture such as SANDS allows an organization to be more responsive to user feedback. For example, if users report that they are extremely satisfied with the user interface of a new clinical system, but very dissatisfied with the accuracy of the drug interaction alerts that the system produces, it would be easy to keep the same clinical system, but substitute a different drug interaction service. It would even be possible to conduct trials of various decision support services to determine which service provides the greatest accuracy and user satisfaction. Modularity would be much more difficult to achieve with other decision support architectures.

7.7 LESSONS ON SERVICE RELIABILITY

One significant risk of a service-oriented architecture is the external dependencies it creates. It is challenging to maintain good performance and uptime in any clinical system, but with an architecture like SANDS, where decision support services are distributed around the country or world, the intensity of the problem is magnified: now reliable operation of the system depends not only on a hospital's ability to keep their own systems and network running, but also on the ability of external service providers to keep their services and network running reliably. The SANDS architecture is informed by significant lessons on reliability from other fields where service-oriented architectures are more prevalent.

Like many problems, service reliability has both technical and organizational dimensions. It is technically feasible to deploy reliable services with excellent uptime properties, as evidenced both here, in section 6.3, and also by the fact that a variety of industries, ranging from banking and insurance to telecommunications and transportation use service-oriented architectures for their mission critical systems. A detailed discussion of technical approaches for achieving high-reliability in service-oriented architectures is outside the scope of this discussion, but techniques such as mirroring, peering, redundancy and quality of service management all contribute to the technical foundation of high-reliability service-oriented architectures, and are well-discussed in the literature (191-198).

I encountered some interesting performance-related issues while prototyping the SANDS architecture, as described in Section 6.5. One that stands out, in particular, is the performance of the Yahoo geocoding service. Its performance, overall, was worse than the equivalent Google service, and there were periodic spikes where its performance dropped substantially. Time-series analysis indicated that the service exhibits diurnal periodicity, as shown in Figure 26. The response time is elevated from 7am until 5pm Pacific Time, peeking in the noon hour. 88% of the queries that took greater than one second to return happen in this time period. The implication is that Yahoo does not have sufficiently capacity to maintain its best performance level under the loads encountered during the daytime hours.

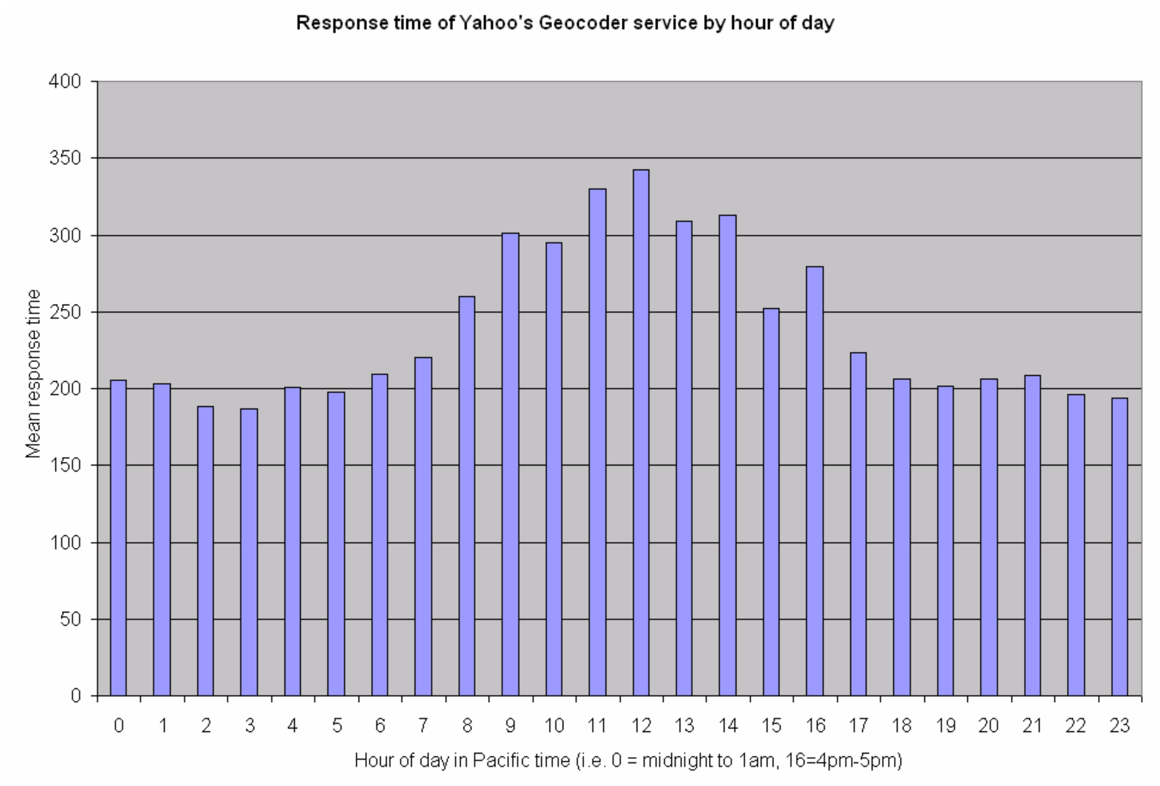


FIGURE 26 - RESPONSE TIME OF YAHOO'S GEOCODER SERVICE BY HOUR OF DAY.

Beyond the technical foundations of reliability lie the organizational foundations. First, it is important for any service consumer to be an informed consumer. Before contracting with a decision support service provider, a consumer should gather data about the reliability of the various alternatives, whether from direct observation (as we did in section 6.3), from current clients of those service providers or from some other data provider, such as a ratings service. Second, reliability should be part of any negotiations with a service provider – just as hospitals are used to negotiating service level agreements (SLA's) with guaranteed minimum reliability thresholds during the purchase of things like telecommunications services, they should also negotiate reliability guarantees with decision support service providers. This aligns the interests of the service consumers with the service providers: unless an acceptable level of reliability is achieved, no money changes hands.

Finally, over time, we might hope that reliability and uptime would become a basis for competition in the market for decision support services. It would be a natural basis for two reasons: first, it is important to consumers of a service, and second, it is a basis for differentiation in the marketplace: just as we saw significant differences across the reliability of three providers of geocoding services in section 6.3, so too might we also expect to see differentiation across other service providers.

7.8 A VISION FOR THE FUTURE OF EHR'S

Developing the SANDS prototype client provided some interesting insight into the role of an electronic health record in an NHIN. Right now, EHR's are mostly viewed as repositories for patient information – the record a provider keeps for any given patient should, ideally, contain all the information needed both to document treatment provided, and to treat that patient in the future. Because today's EHR's are generally not connected, special challenges arise when a patient is being cared for by more than one provider. While the most significant issue this leads to is incomplete information, with all the attendant risks, I found that it also leads to duplicate entry, which often makes it difficult to decipher clinical intent. Consider, for example, a patient who is being seen by two physicians, both of whom use electronic health records, and whose records are not connected. Now, assume that a review of both these records reveals that one physician added Coumadin to the patient's medication list on April 1, and the second added warfarin (the same drug) on April 15. We can reasonably infer that the intent of the first physician was both to prescribe the drug, and document that the patient was taking it. The intent of the second clinician was less clear – was he simply documenting the patients new medication (a good practice), or was he prescribing duplicative (and possibly very dangerous) drug therapy? Because of issues such as this, merging medication lists proved very challenging in development of the SANDS architecture. This is discussed in more detail in Section 7.1, but, fundamentally, it turns out that the practice of duplicate entry, which providers are employing precisely because their systems are not interoperable, actually interferes with the aim of achieving interoperability by making it difficult to integrate and de-duplicate various data sources.

The SANDS client takes a unique approach: unlike traditional EHR's, it has no internal data store – its data store is the NHIN. The data elements one normally finds in an electronic medical record are all present, but because they are read from the network, the need for duplicate entry is obviated – if a provider anywhere orders a drug for a patient, providers everywhere can see it when they view that patient's medication list. I believe that systems like this are the future of clinical systems: in the future, systems ranging from EHR's and CPOE to PHR's and pharmacy systems will be nothing more than purpose-tailored views of a virtual medical record assembled from distributed data sources. While individual end clinical systems might choose, for a variety of reasons such as speed or liability, to mirror data locally, the definitive medical record for a patient would be the union of all records and sources of information regarding that patient made available over the NHIN.   This will, of course, necessitate significant evolution of both our laws and technology, but current efforts such as RHIO's, NHIN prototypes and HHS's Health Information Security and Privacy Collaborative are laying the foundation for these changes.

CHAPTER 8: CONCLUSIONS

8.1 THE CONTRIBUTION OF SANDS

As discussed in Chapter 6 and throughout the dissertation, SANDS has many advantages over other architectures for sharing decision support content. But beyond these technical advantages, the SANDS architecture and the work of this dissertation lay a broader foundation of contributions, including:

- A new way of thinking about electronic health records, not as siloed repositories, but instead as views into connected, distributed data sources.
- Guidance for developing regional and national health information exchange systems which go beyond simple exchange of information by directly enabling clinical decision support and quality measurement.
- An architecture which ties progress on decision support to progress on interoperability. Most other architectures for decision support focus on knowledge representation. The problem of knowledge representation is interesting, but progress on it has been very slow. With SANDS, knowledge representation is no longer a concern, because only the interface to the knowledge is relevant. And SANDS bases that interface on the interfaces (including format and vocabulary standards) currently being developed by HITSP, the NHIN prototypes and others – robust, well-funded and well-supported efforts. With SANDS, as progress is made on interoperability standards progress on decision support will directly follow.

- Better alignment of incentives for decision support by separating responsibilities. With SANDS, the burden of developing decision support no longer falls on the hospital or provider, but can be spread across a variety of entities such as medical specialty societies, government agencies, researchers and commercial interests, each of whom may have a more natural expertise and interest in providing this content than individual providers.

Beyond the SANDS architecture itself, the other key intellectual contribution of this dissertation is the taxonomy of clinical decision support described in Chapter 3. In many ways, this taxonomy is the core theoretical contribution of this dissertation (with the SANDS architecture being the core applied contribution). A taxonomy such as this has broad applicability, not only for developers of clinical decisions support, but also for developers of clinical systems in general, standards development organizations, certifying bodies and policymakers. While prior taxonomies of clinical decision support existed, none was targeted at the functional aspects, and none had as broad an empirical base as this one.

8.2 FINAL NOTES ON ADOPTION AND AVAILABILITY

It is my hope that the SANDS architecture will be an enabling step towards wider adoption of clinical decision support. To that end, I am making the full specifications of the architecture freely available, and am simultaneously releasing a collection of open source tools, libraries and reference implementations to guide those who wish to implement SANDS. While I foresee a wide variety of business models developing around SANDS, I believe that no single person or entity should try to own, control or restrict access to such a network, just as no entity owns or controls the Internet. To that end, I pledge not to seek any exclusionary intellectual property rights to SANDS, such as patents, which would allow me to or others to exert such control. Instead, I will work through open standards processes to ensure that anyone who wishes to access SANDS, either as a provider or consumer of decision support services, can do so freely.

APPENDIX A: FORMAL SERVICE DEFINITION AND SCHEMA

The SANDS service framework is based on XML Web Services and SOAP. Calling a decision support service has three aspects:

1. The client (usually a clinical system) invokes a decision support service by sending it a SOAP message.
2. The service uses the information provided by the client in its invocation, along with other patient information retrieved from the NHIN to make an inference.
3. The service returns an XML-formatted response to the client. This response is defined by SANDS, and validates against the XML schema described in this Appendix.

This Appendix is formal documentation describing the XML schema. Appendix B is a more descriptive reference that describes the schema in more general terms.

Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="decisionSupportResponse">
    <xs:complexType>
      <xs:sequence maxOccurs="unbounded" minOccurs="0">
        <xs:element ref="log" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element ref="showURL"  maxOccurs="unbounded" minOccurs="0"/>
        <xs:element ref="showOrderset"  maxOccurs="unbounded" minOccurs="0"/>
        <xs:element ref="showDataEntryTemplate"  maxOccurs="unbounded" minOccurs="0"/>
        <xs:element ref="getApproval"  maxOccurs="unbounded" minOccurs="0"/>
        <xs:element ref="providePicklist" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element ref="notify" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="engine" use="required" type="xs:string"/>
      <xs:attribute name="ruleset" use="required" type="xs:string"/>
      <xs:attribute name="time" use="required" type="xs:dateTime"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="log" type="xs:string"/>
  <xs:element name="showURL">
    <xs:complexType>
      <xs:attribute name="URL" use="required" type="xs:anyURI"/>
    <xs:attribute name="URLText" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="showOrderset">
  <xs:complexType mixed="true">
    <xs:attribute name="URL" type="xs:anyURI" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="showDataEntryTemplate">
  <xs:complexType mixed="true">
    <xs:attribute name="URL" type="xs:anyURI" use="optional"/>
  </xs:complexType>
</xs:element>
<xs:element name="getApproval">
  <xs:complexType>
    <xs:sequence maxOccurs="unbounded" minOccurs="0">
      <xs:element ref="approvingPerson" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="approvalQueue" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="message" use="required"/>
  </xs:complexType>
</xs:element>
<xs:element name="approvingPerson" type="xs:string"/>
<xs:element name="approvalQueue" type="xs:string"/>
<xs:element name="providePicklist">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="picklistItem"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="picklistItem">
  <xs:complexType>
    <xs:sequence maxOccurs="1" minOccurs="1">
      <xs:element ref="name" maxOccurs="1" minOccurs="1"/>
      <xs:any maxOccurs="unbounded" minOccurs="0" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="name" type="xs:string"/>
<xs:element name="notify">
  <xs:complexType>
    <xs:sequence maxOccurs="unbounded" minOccurs="0">
      <xs:element ref="addAllergy" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="addProblem" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="enterReason" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="writeOrder" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="cancelCurrentOrder" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="cancelExistingOrder" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="deferNotification" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="editCurrentOrder" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="editExistingOrder" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="keepCurrentOrder" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="removeProblem" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="writeLetter" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="writeNote" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="enterAge" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="enterWeight" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="enterHeight" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="guid" use="optional" type="xs:string"/>
    <xs:attribute name="severity" use="required" type="severity"/>
    <xs:attribute name="text" use="required"/>
  </xs:complexType>
</xs:element>
<xs:element name="addAllergy">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="allergenCodeSystem" use="required" type="xs:string"/>
        <xs:attribute name="choiceID" use="required" type="xs:integer"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="addProblem">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:integer">
        <xs:attribute name="choiceID" use="required" type="xs:integer"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="enterReason">
  <xs:complexType mixed="true">
    <xs:attribute name="choiceID" use="required" type="xs:integer"/>
  </xs:complexType>
</xs:element>
<xs:element name="writeOrder">
  <xs:complexType>
    <xs:sequence>
      <xs:any maxOccurs="unbounded" minOccurs="0" />
    </xs:sequence>
    <xs:attribute name="choiceID" use="required" type="xs:integer"/>
  </xs:complexType>
</xs:element>
<xs:element name="number" type="xs:integer"/>
<xs:element name="cancelCurrentOrder">
  <xs:complexType>
    <xs:attribute name="choiceID" use="required" type="xs:integer"/>
  </xs:complexType>
</xs:element>
<xs:element name="cancelExistingOrder">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:integer">
        <xs:attribute name="choiceID" use="required" type="xs:integer"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="deferNotification">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="deferTime"/>
    </xs:sequence>
    <xs:attribute name="choiceID" use="required" type="xs:integer"/>
  </xs:complexType>
</xs:element>
<xs:element name="deferTime" type="xs:duration"/>
<xs:element name="editCurrentOrder">
  <xs:complexType>
    <xs:attribute name="choiceID" use="required" type="xs:integer"/>
  </xs:complexType>
</xs:element>
<xs:element name="editExistingOrder">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:integer">
        <xs:attribute name="choiceID" use="required" type="xs:integer"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="keepCurrentOrder">
  <xs:complexType>
    <xs:attribute name="choiceID" use="required" type="xs:integer"/>
  </xs:complexType>
</xs:element>
<xs:element name="removeProblem">
  <xs:complexType>
```

```
      <xs:simpleContent>
        <xs:extension base="xs:integer">
          <xs:attribute name="choiceID" use="required" type="xs:integer"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="writeLetter">
    <xs:complexType mixed="true">
      <xs:attribute name="choiceID" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="writeNote">
    <xs:complexType mixed="true">
      <xs:attribute name="choiceID" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="enterAge">
    <xs:complexType>
      <xs:attribute name="choiceID" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="enterWeight">
    <xs:complexType>
      <xs:attribute name="choiceID" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="enterHeight">
    <xs:complexType>
      <xs:attribute name="choiceID" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="orderable" type="xs:string"/>
  <xs:element name="dose" type="xs:double"/>
  <xs:element name="doseUnit" type="xs:string"/>
  <xs:element name="frequency" type="xs:string"/>
  <xs:simpleType name="severity">
    <xs:restriction base="xs:integer">
      <xs:enumeration value="1">
        <xs:annotation>
          <xs:documentation>
            Severe - requires immediate, preemptive notification.
          </xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="2">
        <xs:annotation>
          <xs:documentation>
            Moderate - User should be notified, but notification does not need to be
preemptive..
          </xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="3">
        <xs:annotation>
          <xs:documentation>
            Informational - Notification should be provided, but can be low-profile.
          </xs:documentation>
        </xs:annotation>
      </xs:enumeration>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

Schema Document Properties

Target Namespace	None
Element and Attribute Namespaces	• Global element and attribute declarations belong to this schema's target namespace. • By default, local element declarations belong to this schema's target namespace. • By default, local attribute declarations have no namespace.

Declared Namespaces

Prefix	Namespace
xml	http://www.w3.org/XML/1998/namespace
xs	http://www.w3.org/2001/XMLSchema

Schema Component Representation

```
<xs:schema elementFormDefault="qualified">
...
</xs:schema>
```

Global Declarations

Element: addAllergy

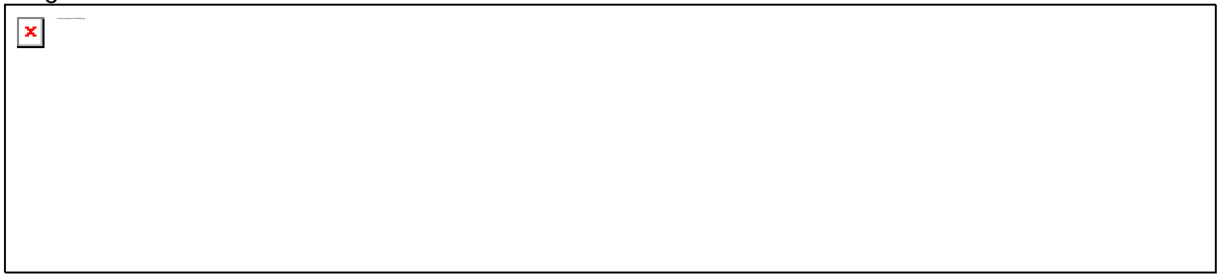
Name	addAllergy
Used by (from the same schema document)	Element **notify**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation

<addAllergy
allergenCodeSystem=" xs:string [1]"
choiceID=" xs:integer [1]">
xs:string
</addAllergy>

Diagram



Schema Component Representation

<xs:element name="addAllergy">
<xs:complexType>
<xs:simpleContent>
<xs:extension base=" xs:string ">
<xs:attribute name="allergenCodeSystem" type=" xs:string " use="required"/>
<xs:attribute name="choiceID" type=" xs:integer " use="required"/>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>

Element: addProblem

Name	addProblem
Used by (from the same schema document)	Element **notify**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<addProblem
choiceID=" xs:integer [1]">
xs:integer
</addProblem>
Diagram

Schema Component Representation
<xs:element name="addProblem">
<xs:complexType>
<xs:simpleContent>
<xs:extension base=" xs:integer ">
<xs:attribute name="choiceID" type=" xs:integer " use="required"/>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>

Element: approvalQueue

Name	approvalQueue
Used by (from the same schema document)	Element **getApproval**
Type	xs:string
Nillable	no
Abstract	no

Logical Diagram

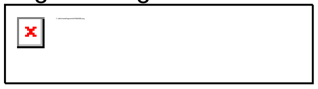
XML Instance Representation
<approvalQueue> xs:string </approvalQueue>
Diagram

Schema Component Representation
<xs:element name="approvalQueue" type=" xs:string "/>

Element: approvingPerson

Name	approvingPerson
Used by (from the same schema document)	Element **getApproval**
Type	xs:string
Nillable	no
Abstract	no

Logical Diagram



XML Instance Representation
<approvingPerson> xs:string </approvingPerson>
Diagram

Schema Component Representation
<xs:element name="approvingPerson" type=" xs:string "/>

Element: cancelCurrentOrder

Name	cancelCurrentOrder
Used by (from the same schema document)	Element **notify**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<cancelCurrentOrder
choiceID=" xs:integer [1]"/>
Diagram

Schema Component Representation
<xs:element name="cancelCurrentOrder">
<xs:complexType>
<xs:attribute name="choiceID" type=" xs:integer " use="required"/>
</xs:complexType>
</xs:element>

Element: cancelExistingOrder

Name	cancelExistingOrder
Used by (from the same schema document)	Element **notify**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<cancelExistingOrder
choiceID=" xs:integer [1]">
xs:integer
</cancelExistingOrder>
Diagram

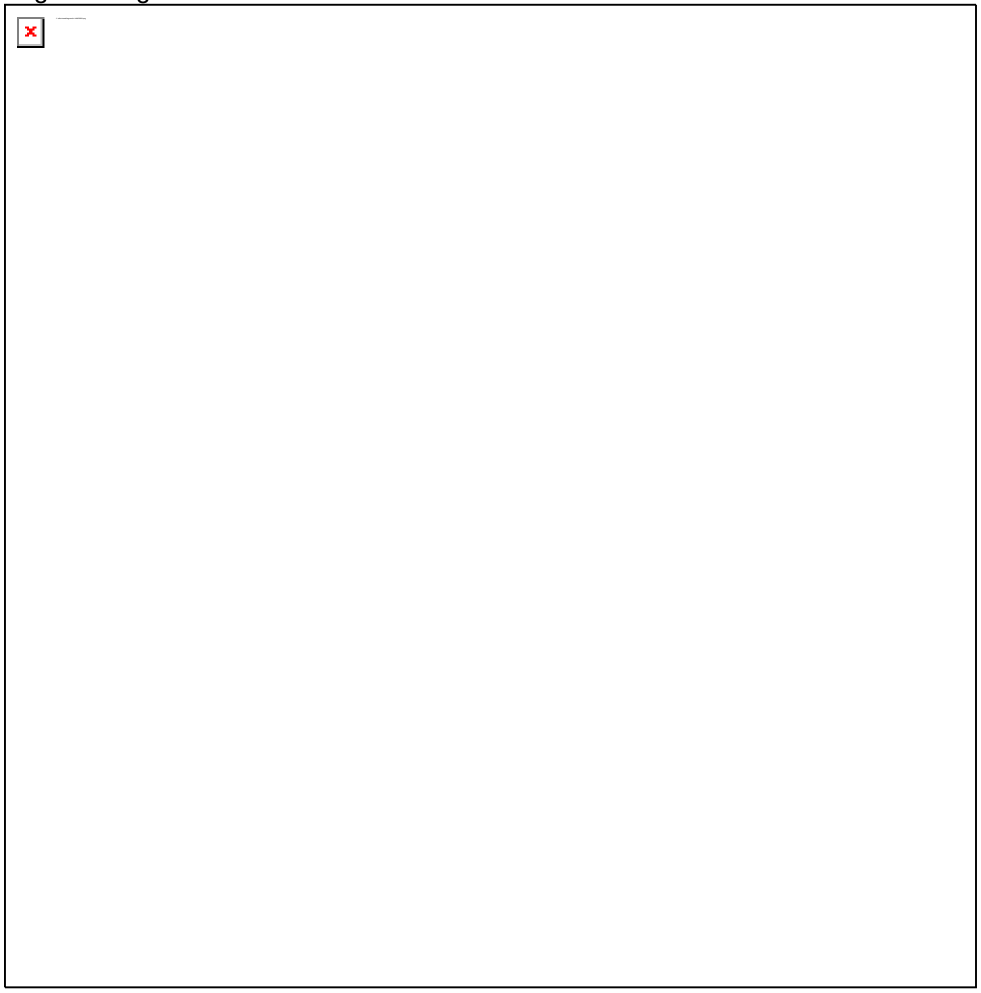
Schema Component Representation
<xs:element name="cancelExistingOrder">
<xs:complexType>
<xs:simpleContent>
<xs:extension base=" xs:integer ">

```
<xs:attribute name="choiceID" type=" xs:integer " use="required"/>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
```

Element: decisionSupportResponse

Name	decisionSupportResponse
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram



XML Instance Representation

```
<decisionSupportResponse
engine=" xs:string [1]"
ruleset=" xs:string [1]"
time=" xs:dateTime [1]">
Start Sequence [0..*]
<log> ... </log> [0..*]
<showURL> ... </showURL> [0..*]
<showOrderset> ... </showOrderset> [0..*]
<showDataEntryTemplate> ... </showDataEntryTemplate> [0..*]
<getApproval> ... </getApproval> [0..*]
<providePicklist> ... </providePicklist> [0..*]
<notify> ... </notify> [0..*]
End Sequence
</decisionSupportResponse>
```

Diagram

Schema Component Representation

```
<xs:element name="decisionSupportResponse">
<xs:complexType>
<xs:sequence maxOccurs="unbounded" minOccurs="0">
<xs:element ref=" log " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" showURL " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" showOrderset " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" showDataEntryTemplate " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" getApproval " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" providePicklist " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" notify " maxOccurs="unbounded" minOccurs="0"/>
</xs:sequence>
```

```
<xs:attribute name="engine" type=" xs:string " use="required"/>
<xs:attribute name="ruleset" type=" xs:string " use="required"/>
<xs:attribute name="time" type=" xs:dateTime " use="required"/>
</xs:complexType>
</xs:element>
```

Element: deferNotification

Name	deferNotification
Used by (from the same schema document)	Element **notify**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation

```
<deferNotification
choiceID=" xs:integer [1]">
<deferTime> ... </deferTime> [1..*]
</deferNotification>
```

Diagram

Schema Component Representation

```
<xs:element name="deferNotification">
<xs:complexType>
<xs:sequence>
<xs:element ref=" deferTime " maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="choiceID" type=" xs:integer " use="required"/>
</xs:complexType>
</xs:element>
```

Element: deferTime

Name	deferTime
Used by (from the same schema document)	Element **deferNotification**
Type	xs:duration
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<deferTime> xs:duration </deferTime>
Diagram

Schema Component Representation
<xs:element name="deferTime" type=" xs:duration "/>

Element: dose

Name	dose
Type	xs:double
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<dose> xs:double </dose>
Diagram

Schema Component Representation
<xs:element name="dose" type=" xs:double "/>

Element: doseUnit

Name	doseUnit
Type	xs:string
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<doseUnit> xs:string </doseUnit>
Diagram

Schema Component Representation
<xs:element name="doseUnit" type=" xs:string "/>

Element: editCurrentOrder

Name	editCurrentOrder
Used by (from the same schema document)	Element **notify**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<editCurrentOrder
choiceID=" xs:integer [1]"/>
Diagram
Schema Component Representation
<xs:element name="editCurrentOrder">
<xs:complexType>
<xs:attribute name="choiceID" type=" xs:integer " use="required"/>
</xs:complexType>
</xs:element>

Element: editExistingOrder

Name	editExistingOrder
Used by (from the same schema document)	Element **notify**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram
XML Instance Representation
<editExistingOrder
choiceID=" xs:integer [1]">
xs:integer
</editExistingOrder>
Diagram
Schema Component Representation
<xs:element name="editExistingOrder">
<xs:complexType>
<xs:simpleContent>
<xs:extension base=" xs:integer ">

<xs:attribute name="choiceID" type=" xs:integer " use="required"/>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>

Element: enterAge

Name	enterAge
Used by (from the same schema document)	Element **notify**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<enterAge
choiceID=" xs:integer [1]"/>
Diagram

Schema Component Representation
<xs:element name="enterAge">
<xs:complexType>
<xs:attribute name="choiceID" type=" xs:integer " use="required"/>
</xs:complexType>
</xs:element>

Element: enterHeight

Name	enterHeight
Used by (from the same schema document)	Element **notify**
Type	Locally-defined complex type

Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<enterHeight
choiceID=" xs:integer [1]"/>
Diagram

Schema Component Representation
<xs:element name="enterHeight">
<xs:complexType>
<xs:attribute name="choiceID" type=" xs:integer " use="required"/>
</xs:complexType>
</xs:element>

Element: enterReason

Name	enterReason
Used by (from the same schema document)	Element **notify**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<enterReason
choiceID=" xs:integer [1]"/>
Diagram

Schema Component Representation
<xs:element name="enterReason">
<xs:complexType mixed="true">

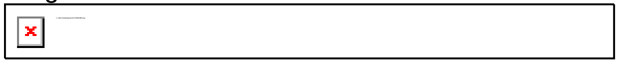
<xs:attribute name="choiceID" type=" xs:integer " use="required"/>
</xs:complexType>
</xs:element>

Element: enterWeight

Name	enterWeight
Used by (from the same schema document)	Element **notify**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<enterWeight
choiceID=" xs:integer [1]"/>
Diagram



Schema Component Representation
<xs:element name="enterWeight">
<xs:complexType>
<xs:attribute name="choiceID" type=" xs:integer " use="required"/>
</xs:complexType>
</xs:element>

Element: frequency

Name	frequency
Type	xs:string
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<frequency> xs:string </frequency>
Diagram

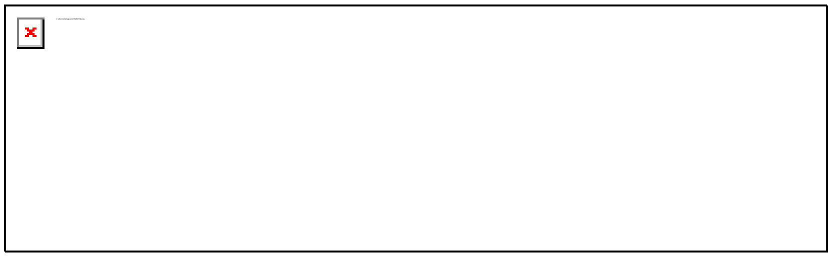
Schema Component Representation
<xs:element name="frequency" type=" xs:string "/>

Element: getApproval

Name	getApproval
Used by (from the same schema document)	Element **decisionSupportResponse**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<getApproval
message="anySimpleType [1]">
Start Sequence [0..*]
<approvingPerson> ... </approvingPerson> [0..*]
<approvalQueue> ... </approvalQueue> [0..*]
End Sequence
</getApproval>
Diagram



Schema Component Representation

```
<xs:element name="getApproval">
<xs:complexType>
<xs:sequence maxOccurs="unbounded" minOccurs="0">
<xs:element ref=" approvingPerson " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" approvalQueue " maxOccurs="unbounded" minOccurs="0"/>
</xs:sequence>
<xs:attribute name="message" use="required"/>
</xs:complexType>
</xs:element>
```

Element: keepCurrentOrder

Name	keepCurrentOrder
Used by (from the same schema document)	Element **notify**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation

```
<keepCurrentOrder
choiceID=" xs:integer [1]"/>
```

Diagram

Schema Component Representation

```
<xs:element name="keepCurrentOrder">
<xs:complexType>
<xs:attribute name="choiceID" type=" xs:integer " use="required"/>
</xs:complexType>
</xs:element>
```

Element: log

Name	log
Used by (from the same schema document)	Element **decisionSupportResponse**
Type	xs:string
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<log> xs:string </log>
Diagram

Schema Component Representation
<xs:element name="log" type=" xs:string "/>

Element: name

Name	name
Used by (from the same schema document)	Element **picklistItem**
Type	xs:string
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation

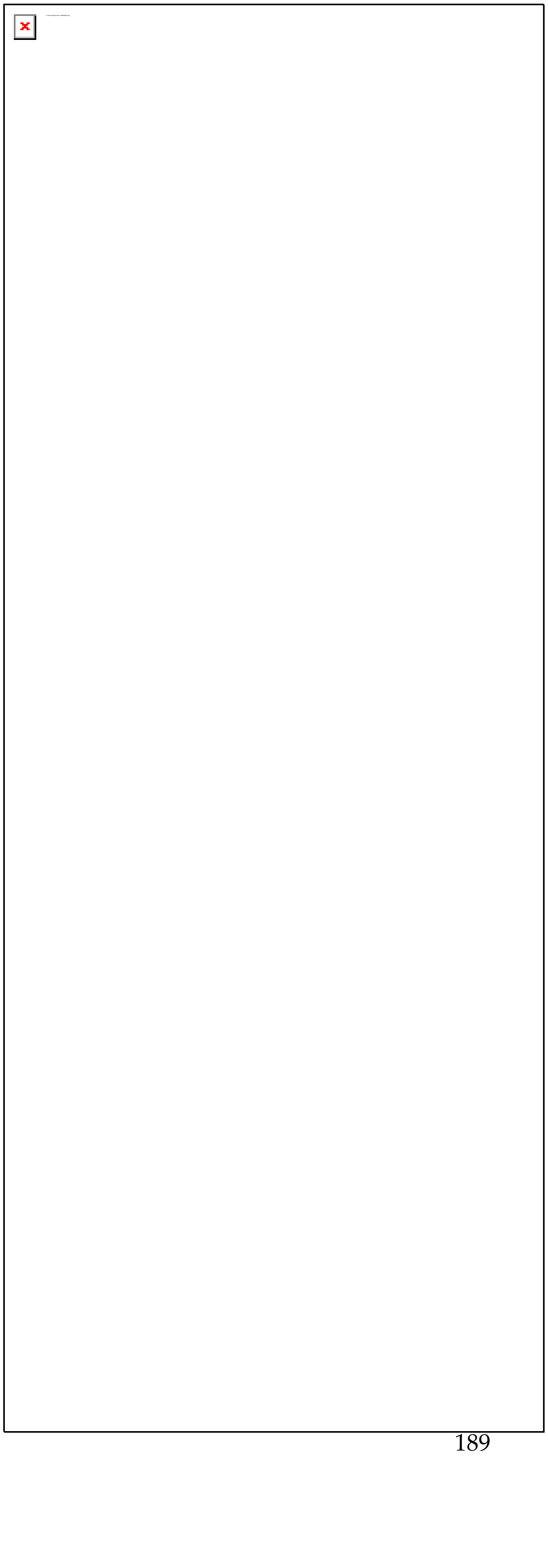
<name> xs:string </name>
Diagram

Schema Component Representation
<xs:element name="name" type=" xs:string "/>

Element: notify

Name	notify
Used by (from the same schema document)	Element **decisionSupportResponse**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram



XML Instance Representation
<notify
guid=" xs:string [0..1]"
severity=" severity [1]"
text="anySimpleType [1]">
Start Sequence [0..*]
<addAllergy> ... </addAllergy> [0..*]
<addProblem> ... </addProblem> [0..*]
<enterReason> ... </enterReason> [0..*]
<writeOrder> ... </writeOrder> [0..*]
<cancelCurrentOrder> ... </cancelCurrentOrder> [0..*]
<cancelExistingOrder> ... </cancelExistingOrder> [0..*]
<deferNotification> ... </deferNotification> [0..*]
<editCurrentOrder> ... </editCurrentOrder> [0..*]
<editExistingOrder> ... </editExistingOrder> [0..*]
<keepCurrentOrder> ... </keepCurrentOrder> [0..*]
<removeProblem> ... </removeProblem> [0..*]
<writeLetter> ... </writeLetter> [0..*]
<writeNote> ... </writeNote> [0..*]
<enterAge> ... </enterAge> [0..*]
<enterWeight> ... </enterWeight> [0..*]
<enterHeight> ... </enterHeight> [0..*]
End Sequence
</notify>
Diagram

Schema Component Representation

```
<xs:element name="notify">
<xs:complexType>
<xs:sequence maxOccurs="unbounded" minOccurs="0">
<xs:element ref=" addAllergy " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" addProblem " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" enterReason " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" writeOrder " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" cancelCurrentOrder " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" cancelExistingOrder " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" deferNotification " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" editCurrentOrder " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" editExistingOrder " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" keepCurrentOrder " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" removeProblem " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" writeLetter " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" writeNote " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" enterAge " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" enterWeight " maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref=" enterHeight " maxOccurs="unbounded" minOccurs="0"/>
</xs:sequence>
<xs:attribute name="guid" type=" xs:string " use="optional"/>
<xs:attribute name="severity" type=" severity " use="required"/>
<xs:attribute name="text" use="required"/>
</xs:complexType>
</xs:element>
```

Element: number

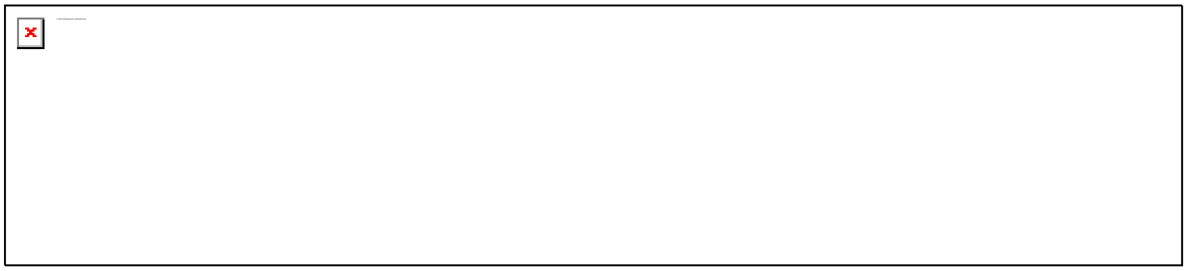
Name	number
Type	xs:integer
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation

```
<number> xs:integer </number>
```

Diagram



Schema Component Representation
<xs:element name="number" type=" xs:integer "/>

Element: orderable

Name	orderable
Type	xs:string
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<orderable> xs:string </orderable>
Diagram

Schema Component Representation
<xs:element name="orderable" type=" xs:string "/>

Element: picklistItem

Name	picklistItem
Used by (from the same schema document)	Element **providePicklist**
Type	Locally-defined complex type
Nillable	no

Abstract	no

Logical Diagram

XML Instance Representation
<picklistItem>
<name> ... </name> [1]
Allow any elements from any namespace (strict validation). [0..*]
</picklistItem>
Diagram

Schema Component Representation
<xs:element name="picklistItem">
<xs:complexType>
<xs:sequence maxOccurs="1" minOccurs="1">
<xs:element ref=" name " maxOccurs="1" minOccurs="1"/>
<xs:any maxOccurs="unbounded" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>

Element: providePicklist

Name	providePicklist
Used by (from the same schema document)	Element **decisionSupportResponse**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<providePicklist>
<picklistItem> ... </picklistItem> [1..*]

```
</providePicklist>
```

Diagram

Schema Component Representation

```
<xs:element name="providePicklist">
<xs:complexType>
<xs:sequence>
<xs:element ref=" picklistItem " maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

Element: removeProblem

Name	removeProblem
Used by (from the same schema document)	Element **notify**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation

```
<removeProblem
choiceID=" xs:integer [1]">
xs:integer
</removeProblem>
```

Diagram

Schema Component Representation

```
<xs:element name="removeProblem">
<xs:complexType>
<xs:simpleContent>
<xs:extension base=" xs:integer ">
<xs:attribute name="choiceID" type=" xs:integer " use="required"/>
</xs:extension>
```

```
</xs:simpleContent>
</xs:complexType>
</xs:element>
```

Element: showDataEntryTemplate

Name	showDataEntryTemplate
Used by (from the same schema document)	Element **decisionSupportResponse**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation

```
<showDataEntryTemplate
URL=" xs:anyURI [0..1]"/>
```

Diagram

Schema Component Representation

```
<xs:element name="showDataEntryTemplate">
<xs:complexType mixed="true">
<xs:attribute name="URL" type=" xs:anyURI " use="optional"/>
</xs:complexType>
</xs:element>
```

Element: showOrderset

Name	showOrderset
Used by (from the same schema document)	Element **decisionSupportResponse**
Type	Locally-defined complex type

Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<showOrderset
URL=" xs:anyURI [0..1]"/>
Diagram

Schema Component Representation
<xs:element name="showOrderset">
<xs:complexType mixed="true">
<xs:attribute name="URL" type=" xs:anyURI " use="optional"/>
</xs:complexType>
</xs:element>

Element: showURL

Name	showURL
Used by (from the same schema document)	Element **decisionSupportResponse**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<showURL
URL=" xs:anyURI [1]"
URLText="anySimpleType [0..1]"/>
Diagram

Schema Component Representation
<xs:element name="showURL">
<xs:complexType>
<xs:attribute name="URL" type=" xs:anyURI " use="required"/>
<xs:attribute name="URLText" use="optional"/>
</xs:complexType>
</xs:element>

Element: writeLetter

Name	writeLetter
Used by (from the same schema document)	Element **notify**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<writeLetter
choiceID=" xs:integer [1]"/>
Diagram

Schema Component Representation
<xs:element name="writeLetter">
<xs:complexType mixed="true">
<xs:attribute name="choiceID" type=" xs:integer " use="required"/>
</xs:complexType>
</xs:element>

Element: writeNote

Name	writeNote
Used by (from the same schema document)	Element **notify**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<writeNote
choiceID=" xs:integer [1]"/>
Diagram

Schema Component Representation
<xs:element name="writeNote">
<xs:complexType mixed="true">
<xs:attribute name="choiceID" type=" xs:integer " use="required"/>
</xs:complexType>
</xs:element>

Element: writeOrder

Name	writeOrder
Used by (from the same schema document)	Element **notify**
Type	Locally-defined complex type
Nillable	no
Abstract	no

Logical Diagram

XML Instance Representation
<writeOrder
choiceID=" xs:integer [1]">
Allow any elements from any namespace (strict validation). [0..*]
</writeOrder>
Diagram

Schema Component Representation
<xs:element name="writeOrder">
<xs:complexType>
<xs:sequence>
<xs:any maxOccurs="unbounded" minOccurs="0"/>
</xs:sequence>
<xs:attribute name="choiceID" type=" xs:integer " use="required"/>
</xs:complexType>
</xs:element>

Global Definitions

Simple Type: severity

Super-types:	xs:integer < **severity** (by restriction)
Sub-types:	None

Name	severity
Used by (from the same schema document)	Element **notify**
Content	• Base XSD Type: integer • *value* comes from list: {'1'|'2'|'3'}

Diagram

Schema Component Representation

```
<xs:simpleType name="severity">
<xs:restriction base=" xs:integer ">
<xs:enumeration value="1"/>
<xs:enumeration value="2"/>
<xs:enumeration value="3"/>
</xs:restriction>
</xs:simpleType>
```

APPENDIX B: GUIDE TO THE SERVICE DEFINITION

INTRODUCTION

While Appendix A gave a formal schema and description of the SANDS response message format, this appendix is designed to give a more user-friendly description of the response message types. It is perhaps easiest to introduce the response message format with some examples: first, a response to a preventive health query, which suggests a statin for a diabetic patient with insufficiently controlled LDL (and is based on a rule in use right now at Partners HealthCare), and second, an empty response, when the engine has nothing to suggest. More examples are given in the proof-of-concept mappings, included here as Appendix A.

Drug Suggestion for a patient with DM and hyperlipidemia:

```
<decisionSupportResponse engine="TestEngine"
                         ruleset="OutpatientVisit"
                         time="2007-01-31T12:12:05-05:00">
    <notify severity="3"
            text="Patient has diabetes and most recent LDL (144) above target (100) and
                  not on an HMG CoA-reductase inhibitor.  Recommend HMG CoA-reductase
                  inhibitor if not allergic."
            guid="35b6c020-3e95-11db-a98b-0800200c9a66">
       <writeOrder choiceID="1">
          <rxcui>36567</rxcui> <!--Simvastatin-->
          <dose>20</dose>
          <doseUnit>mg</doseUnit>
          <frequency>qd</frequency>
          <number>90</number>
       </writeOrder>
       <addAllergy choiceID="2" allergenCodeSystem="mesh">
           D27.505.519.186.071.202.370
       </addAllergy>
       <addProblem choiceID="3">266468003</addProblem>
       <enterReason choiceID="4">Decline suggestion: Other Comorbidities</enterReason>
       <enterReason choiceID="5">Decline suggestion: Cost of Treatment</enterReason>
       <enterReason choiceID="6">Decline suggestion: Patient refuses</enterReason>
       <removeProblem choiceID="7">73211009</removeProblem>
       <deferNotification choiceID="8">
             <deferTime>P5D</deferTime >
             <deferTime>P1M</deferTime >
             <deferTime>P3M</deferTime >
       </deferNotification>
    </notify>
</decisionSupportResponse>
```

Empty response:

```
<decisionSupportResponse engine="TestEngine"
                ruleset="OutpatientVisit"
                        time=" 2006-08-24T12:12:05-05:00"/>
```

RESPONSE PREAMBLE

Each response from the decision support engine is encoded as an XML document, according to an XML schema. The root node of this document is decisionSupportResponse. The attributes of this node form the preamble for the response, and are described in this section. The only legal children of this root node are response actions, which are described in the next section. The three attributes of decisionSupportResponse are:

- Engine: A descriptor of the rule engine. This should ideally encode the name and version of the engine, and may optionally encode the machine the engine is running on, for debugging purposes.
- Ruleset: The name of the ruleset executed. This should match the first argument of the RunRules function used to invoke the service.
- Time: A timestamp, in valid XML datetime format.

RESPONSE ACTIONS

The core of the decision support response is a set of response actions. Each response can have zero or more response actions, which are then rendered by the receiving application, according to the guidance provided in this service definition.

NOTIFY

Description:

The notify action is used to provide information to a user, and optionally to offer choices to the user. These choices are described in the Response Choices section of this service definition, and are children of the `notify` node in the XML schema.

Required Attributes:

- Severity: Severity is coded 1, 2 or 3:
    - 1. Severe: Requires immediate, preemptive notification, such as paging, or displaying a popup.
    - 2. Moderate: User should be notified, but notification does not need to be preemptive.
    - 3. Informational: Notification should be provided, but can be done in a low-profile way.
- Text: Text is a valid XML string (which must be escaped if it contains special characters). This text will be displayed to the user.

Optional Attributes:

- guid: A GUID unique to this notification. If a GUID is provided, the application must provide a response back to the service specifying the GUID and the response action chosen by the user. If no GUID is provided, no response will be returned to the service. The two use cases for this information are variable snooze periods and reporting / data collection. In both these cases, the engine will need to know what action the user chose, and the GUID will enable this.

Example:

```
<notify severity="1"
 text="Vincristine ordered with intrathecal route. This route of
        administration is fatal and absolutely contraindicated.">
```

LOG

Description:

The log action is used to log a piece of information, without providing any notification to the user. It is most frequently used in surveillance rules, and in research.

Element Contents:

The receiving clinical system will log the contents of the XML log element as text. The contents must be properly escaped if it contains special characters.

Example:

```
<log>Rising HGH ordered without documented indication.</log>
```

SHOWURL

Description:

The showURL action shows a URL. Applications may choose to automatically launch the page, show it inline, or present it as a clickable hyperlink. If it is presented as a hyperlink, the URLText attribute should be used.

Required Attributes:

- URL: The URL to be displayed

Optional Attributes:

- URLText: The text to be displayed in the hyperlink. If this attribute is omitted, and the application chooses to display a hyperlink, it should substitute the URL for the text of the hyperlink.

Example:

```
<showURL
  URL="http://www.oqp.med.va.gov/cpg/CHF/28766chronicheartfailure.pdf"
  URLText="VA/DoD CHF Guideline">
```

SHOWORDERSET

Description:

The showOrderset action shows an orderset.

Optional Attributes:

- URL: The URL for an HL7 orderset. The application will fetch that orderset and render it. If this attribute is not specified, the application will interpret the contents of the showOrderset element as an HL7 orderset.

Element Contents:

A valid HL7 orderset to be rendered by the application. The element contents are only read if no URL attribute is specified. If both the URL and element contents are specified, the application will ignore the element contents. This is not an error, but the application may choose to raise a warning.

Example:

```
<showOrderset URL="http://ordersets.org/copd.xml">
```

SHOWDATAENTRYTEMPLATE

Description:

The showDataEntryTemplate action shows a data entry template.

Optional Attributes:

- URL: The URL for an XForms document. The application will fetch that document and render it. If this attribute is not specified, the application will interpret the contents of the element as an XForms document.

Element Contents:

A valid XForms document to be rendered by the application. The element contents are only read if no URL attribute is specified. If both the URL and element contents are specified, the application will ignore the element contents. This is not an error, but the application may choose to raise a warning.

Example:

```
<showDataEntryTemplate
    URL=" http://mozilla.org/projects/xforms/samples/tax_form/TaxForm.xhtml" />
```

GETAPPROVAL

Description:

The getApproval action instructs the clinical system to route the current action to an approving person or an approval queue. This action should not be used for standard order routing workflows – it should only be used when specific clinical circumstances require special approvals.

Element Contents:

The element should contain one or more child elements of type `approvingPerson` or `approvalQueue`. These elements are described as follows:

- approvingPerson: The NPI of the person to whom the order should be routed for approval.
- approvalQueue: An implementation-specific queue identifier.

If more than one approving person or approving queue is included, all of the approvers must provide approval. If a group of approvers is available, and only one of them is required to approve the action, a corresponding approval queue should be created.

Example:

```
<getApproval message="Ordering HGH for non-approved indication">
    <approvingPerson>2739428672</approvingPerson>
    <approvalQueue>Infectious Disease</approvalQueue>
</getApproval>
```

PROVIDEPICKLIST

Description:

The providePicklist action is used to provide items for a picklist, or to provide a default value for use in a UI.

Required Element Contents:

The element should contain one or more child elements of type `picklistItem`. In turn, each `picklistItem` should contain at least one node. The child nodes for `picklistItem` are highly context-dependent, and their selection is left up to the implementer. A `name` element containing a human readable name for the choice is recommended, unless it can be inferred automatically by the application.

This action type can also be used to provide a single default value. This is simply implemented as a case where the `providePicklist` element contains only a single `picklistItem`.

Example:

```
<providePicklist>
    <picklistItem>
          <name>Aspirin</name>
          <rxcui>1191</rxcui>
          <dose>650</dose>
          <doseUnit>mg</doseUnit>
          <frequency>q4h prn</frequency>
       </writeOrder>
    </picklistItem>
    <picklistItem>
          <name>Acetaminophen</name>
          <rxcui>161</rxcui>
          <dose>500</dose>
          <doseUnit>mg</doseUnit>
          <frequency>q4h prn</frequency>
       </writeOrder>
    </picklistItem>
</providePicklist>
```

RESPONSE CHOICES

Response choices are children of the notify response action. A single notify action can contain an unlimited number of choices. Each response choice should have a unique value in its `choiceID` attribute. This, together with the notify action's GUID allow for unique identification of the choice made for a notification, as described in the notify response action section.

WRITEORDER

Description:

The writeOrder choice allows the user to order something.

Element Contents:

The `writeOrder` element is required to contain an `orderable` child element. It may optionally contain other elements, which are item specific, and can be freely extended.

Example:

```
<writeOrder choiceID="1">
      <orderable>327</orderable>
      <dose>500</dose>
      <doseUnit>mg</doseUnit>
      <frequency>q4h prn</frequency>
</writeOrder>
```

DEFERNOTIFICATION

Description:

The deferNotification choice allows the user to defer the notification they received to a later date. The notification and context should be stored by the application, and presented to the user at the selected time. Note that this is different than the snooze functionality that may be implemented by a rule. Snooze does not cause the notification to reappear after a set period of time – it simply stops the notification from being generated during that period of time. It will not be regenerated after the snooze until the rule is triggered again.

Element Contents:

The `deferNotification` element is required to contain one or more `deferTime` child elements. These elements have an XML duration in ISO 8601 format as their contents:

Example:

```
<deferNotification choiceID="1">

      <deferTime>P5D</deferTime >

      <deferTime>P1M</deferTime >

      <deferTime>P3M</deferTime >

</deferNotification>
```

KEEPCURRENTORDER

Description:

The keepCurentOrder choice allows the user to keep the order that prompted the notification action. It is essentially an override. There are no attributes or element contents, other than the standard choiceID attribute.

Example:

```
<keepCurrentOrder choiceID="1"/>
```

CANCELCURRENTORDER

Description:

The cancelCurentOrder choice allows the user to cancel the order that prompted the notification action. When chosen by the user, the application should return to whatever state it was in before the user started the order which triggered the notify action. There are no attributes or element contents, other than the standard choiceID attribute.

Example:

```
<cancelCurrentOrder choiceID="1"/>
```

EDITCURRENTORDER

Description:

The editCurentOrder choice allows the user to edit the order that prompted the notification action. When chosen by the user, the application should allow editing of the order which triggered the notify action. After the order is edited, the engine should be re-invoked as though the user had entered the newly edited order from scratch. There are no attributes or element contents, other than the standard choiceID attribute.

Example:

```
<editCurrentOrder choiceID="1"/>
```

CANCELEXISTINGORDER

Description:

The cancelExistingOrder choice allows the user to cancel an already existing order for a patient. When chosen by the user, the application should cancel the existing order, and continue through the ordering workflow for the current (new) order.

Element Contents:

The contents of the cancelExistingOrder element should be an ID which identifies the existing order to be cancelled.

Example:

```
<cancelExistingOrder choiceID="1">27314</cancelExistingOrder>
```

EDITEXISTINGORDER

Description:

The editExistingOrder choice allows the user to edit an already existing order for a patient. When chosen by the user, the application should suspend the current ordering context, and allow the user to edit the order in question (with full decision support for that ordering session). After the existing order is revised, the current ordering session should be resumed, and the engine should be re-invoked for the current (new) order that triggered the notify action.

Element Contents:

The contents of the editExistingOrder element should be an ID which identifies the existing order to be edited.

Example:

```
<editExistingOrder choiceID="1">27314</editExistingOrder>
```

ADDALLERGY

Description:

The addAllergy choice allows the user to set a new allergy a patient. When chosen by the user, the application should add the allergy to the allergy list and then re-invoke the engine in the context that generated the notify action. The addAllergy response action uses the CHI recommended vocabularies (http://www.ncvhs.hhs.gov/060913lt.htm) for allergen. Only allergen is specified. Systems which need information on severity or reaction should prompt the user for this information.

Required Attributes:

- allergenCodeSystem: The coding system used to describe the allergen. Must be one of `unii`, `rxnormbn`, `ndfrt`, or `mesh`.

Element Contents:

The element should contain the allergen code according to the chosen code system.

Example:

```
<addAllergy choiceID="1" allergenCodeSystem="mesh">

  D27.505.519.186.071.202.370

</addAllergy>
```

Note: `D27.505.519.186.071.202.370` is the MESH term for Hydroxymethylglutaryl-CoA Reductase Inhibitors.

WRITENOTE

Description:

The writeNote choice starts a progress note. If chosen by the user, the progress note should appear in an editor, where it can be revised and finalized..

Element Contents:

The default text for the progress note, which must be properly escaped if it contains special characters.

Example:

```
        <writeNote choiceID="1">I referred the patient to a diabetes educator for
assistance with his glycemic control. </writeNote>
```

WRITELETTER

Description:

The writeLetter choice generates a letter to the patient.

Element Contents:

The body of the letter, which must be properly escaped if it contains special characters. The application will add the appropriate greeting and salutation.

Example:

```
        <writeLetter choiceID="1"> Your most recent hemoglobin A1C test result was 11%.
This is higher than your goal of 7%.  Please schedule an appointment with your physician
or diabetes educator to discuss options for reducing your HbA1c.</writeLetter>
```

ADDPROBLEM

Description:

The addProblem choice adds a problem to the patient's problem list. After the problem is added, the engine should be re-invoked in the same context that generated the notify action.

Element Contents:

The element should contain a SNOMED code for the problem to be added.

Example:

```
        <addProblem choiceID="1">266468003</addProblem>
```

REMOVEPROBLEM

Description:

The removeProblem choice removes a problem from the patient's problem list. After the problem is removed, the engine should be re-invoked in the same context that generated the notify action.

Element Contents:

The element should contain a SNOMED code for the problem to be removed.

Example:

```
<addProblem choiceID="1">13644009</addProblem>
```

ENTERAGE

Description:

The enterAge choice gives the user the option to enter the patient's age. It should only be presented when the age is unknown, or suspected to be inaccurate. If the user chooses this option and enters an age, the ruleset should be invoked again, in the same context as it was previously. There are no attributes or element contents, other than the standard choiceID attribute.

Example:

```
<enterAge choiceID="1" />
```

ENTERWEIGHT

Description:

The enterWeight choice gives the user the option to enter the patient's weight. It should only be presented when the weight is unknown, or suspected to be inaccurate. If the user chooses this option and enters a weight, the ruleset should be invoked again, in the same context as it was previously. There are no attributes or element contents, other than the standard choiceID attribute.

Example:

```
<enterWeight choiceID="1" />
```

ENTERHEIGHT

Description:

The enterHeight choice gives the user the option to enter the patient's height. It should only be presented when the height is unknown, or suspected to be inaccurate. If the user chooses this option and enters a height, the ruleset should be invoked again, in the same context as it was previously. There are no attributes or element contents, other than the standard choiceID attribute.

Example:

```
<enterHeight choiceID="1" />
```

ENTERREASON

Description:

The enterReason choice allows the user to select a reason. This choice is a form of override – after the reason is chosen, the application logs the reason chosen, and proceeds with the action that prompted the notify action continues unmodified. The enterReason choice is relatively unstructured, so other, more specific choices should be chosen where possible.

Element Contents:

The reason to be displayed to the user and logged.

Example:

```
<enterReason choiceID="1">Patient Refused</enterReason>
```

OTHERREASON

Description:

The otherReason choice allows the user to input a free-text reason. This choice is a form of override – after the reason is chosen, the application logs the reason chosen, and proceeds with the action that prompted the notify action continues unmodified. The otherReason choice is relatively unstructured, so other, more specific choices should be chosen where possible. There are no attributes or element contents, other than the standard choiceID attribute.

Example:

```
<otherReason choiceID="1"/>
```

RETURNING THE USER'S CHOICE TO THE SERVICE

In some cases, it is it desirable to return the choice that a user made to the decision support service. This might be useful for statistical purposes, such as determining how frequently a user accepted a suggestion, or which of several choices they selected. It can also be used to cause rules to snooze for a period of time, as described above. When the user makes a choice, the application should make a return call to the service:

```
RegisterResponse(guid, choiceID)
```

Where guid is the GUID attribute of the notify response action, and choiceID is the choiceID attribute of the choice the user made. in the drug Suggestion for a patient with DM and hyperlipidemia described in the Response Overview section, if the patient refused the statin due to cost of treatment, the return call would be:

```
RegisterResponse("35b6c020-3e95-11db-a98b-0800200c9a66", "4")
```

REFERENCES

1. Osheroff JA, Teich JM, Middleton B, Steen EB, Wright A, Detmer DE. A Roadmap for National Action on Clinical Decision Support. 2006.

2. Teich JM, Merchia PR, Schmiz JL, Kuperman GJ, Spurr CD, Bates DW. Effects of computerized physician order entry on prescribing practices. Arch Intern Med. 2000 Oct 9;160(18):2741-7.

3. Teich JM, Spurr CD, Flammini SJ, et al. Response to a trial of physician-based inpatient order entry. Proc Annu Symp Comput Appl Med Care. 1993:316-20.

4. Baorto DM, Cimino JJ. An "infobutton" for enabling patients to interpret on-line Pap smear reports. Proc AMIA Symp. 2000:47-50.

5. Cimino JJ, Li J. Sharing infobuttons to resolve clinicians' information needs. AMIA Annu Symp Proc. 2003:815.

6. Cimino JJ, Li J, Bakken S, Patel VL. Theoretical, empirical and practical approaches to resolving the unmet information needs of clinical information system users. Proc AMIA Symp. 2002:170-4.

7. Chin HL, Krall M. Implementation of a comprehensive computer-based patient record system in Kaiser Permanente's Northwest Region. MD Comput. 1997 Jan-Feb;14(1):41-5.

8. Garrido T, Jamieson L, Zhou Y, Wiesenthal A, Liang L. Effect of electronic health records in ambulatory care: retrospective, serial, cross sectional study. Bmj. 2005 Mar 12;330(7491):581.

9. Osheroff JA, Pifer EA, Sittig DF, Jenders RA, Teich JM. Improving outcomes with clinical decision support: an implementers' guide. Chicago: HIMSS; 2005.

10. Barnett GO, Cimino JJ, Hupp JA, Hoffer EP. DXplain. An evolving diagnostic decision-support system. Jama. 1987 Jul 3;258(1):67-74.

11. de Dombal FT, Leaper DJ, Staniland JR, McCann AP, Horrocks JC. Computer-aided diagnosis of acute abdominal pain. Br Med J. 1972 Apr 1;2(5804):9-13.

12. Gorry GA, Barnett GO. Experience with a model of sequential diagnosis. Comput Biomed Res. 1968 May;1(5):490-507.

13. Gorry GA, Barnett GO. Sequential diagnosis by computer. Jama. 1968 Sep 16;205(12):849-54.

14. Miller RA. Medical diagnostic decision support systems--past, present, and future: a threaded bibliography and brief commentary. J Am Med Inform Assoc. 1994 Jan-Feb;1(1):8-27.

15. Miller RA, Masarie FE, Jr. The demise of the "Greek Oracle" model for medical diagnostic systems. Methods Inf Med. 1990 Jan;29(1):1-2.

16. Miller RA, Pople HE, Jr., Myers JD. Internist-1, an experimental computer-based diagnostic consultant for general internal medicine. N Engl J Med. 1982 Aug 19;307(8):468-76.

17. Nelson SJ, Blois MS, Tuttle MS, et al. Evaluating RECONSIDER. A computer program for diagnostic prompting. J Med Syst. 1985 Dec;9(5-6):379-88.

18. Weiss S, Kulikowski CA, Safir A. Glaucoma consultation by computer. Comput Biol Med. 1978 Jan;8(1):25-40.

19. Balas EA, Weingarten S, Garb CT, Blumenthal D, Boren SA, Brown GD. Improving preventive care by prompting physicians. Arch Intern Med. 2000 Feb 14;160(3):301-8.

20. Cabana MD, Rand CS, Powe NR, et al. Why don't physicians follow clinical practice guidelines? A framework for improvement. Jama. 1999 Oct 20;282(15):1458-65.

21. Garg AX, Adhikari NK, McDonald H, et al. Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: a systematic review. Jama. 2005 Mar 9;293(10):1223-38.

22. Grimshaw JM, Russell IT. Effect of clinical guidelines on medical practice: a systematic review of rigorous evaluations. Lancet. 1993 Nov 27;342(8883):1317-22.

23. Hunt DL, Haynes RB, Hanna SE, Smith K. Effects of computer-based clinical decision support systems on physician performance and patient outcomes: a systematic review. Jama. 1998 Oct 21;280(15):1339-46.

24. Johnston ME, Langton KB, Haynes RB, Mathieu A. Effects of computer-based clinical decision support systems on clinician performance and patient outcome. A critical appraisal of research. Ann Intern Med. 1994 Jan 15;120(2):135-42.

25. Kawamoto K, Houlihan CA, Balas EA, Lobach DF. Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success. Bmj. 2005 Apr 2;330(7494):765.

26. Kohn LT, Corrigan J, Donaldson MS. To err is human: building a safer health system. Washington, D.C.: National Academy Press; 2000.

27. McGlynn EA, Asch SM, Adams J, et al. The quality of health care delivered to adults in the United States. N Engl J Med. 2003 Jun 26;348(26):2635-45.

28. Chaudhry B, Wang J, Wu S, et al. Systematic Review: Impact of Health Information Technology on Quality, Efficiency, and Costs of Medical Care. Ann Intern Med. 2006;144(10).

29. Ledley RS, Lusted LB. Reasoning foundations of medical diagnosis; symbolic logic, probability, and value theory aid our understanding of how physicians reason. Science. 1959 Jul 3;130(3366):9-21.

30. Sittig DF, Ash JS, Ledley RS. The story behind the development of the first whole-body computerized tomography scanner as told by Robert S. Ledley. J Am Med Inform Assoc. 2006 Sep-Oct;13(5):465-9.

31. O'Connor GT, Sox HC, Jr. Bayesian reasoning in medicine: the contributions of Lee B. Lusted, MD. Med Decis Making. 1991 Apr-Jun;11(2):107-11.

32. Warner HR, Toronto AF, Veasey LG, Stephenson R. A mathematical approach to medical diagnosis. Application to congenital heart disease. Jama. 1961 Jul 22;177:177-83.

33. Collen MF, Rubin L, Neyman J, Dantzig GB, Baer RM, Siegelaub AB. Automated Multiphasic Screening And Diagnosis. Am J Public Health Nations Health. 1964 May;54:741-50.

34. Bleich HL. Computer evaluation of acid-base disorders. J Clin Invest. 1969 Sep;48(9):1689-96.

35. de Dombal FT, Horrocks JC, Staniland JR, Guillou PJ. Construction and uses of a 'data-base' of clinical information concerning 600 patients with acute abdominal pain. Proc R Soc Med. 1971 Sep;64(9):978.

36. Shortliffe EH, Davis R, Axline SG, Buchanan BG, Green CC, Cohen SN. Computer-based consultations in clinical therapeutics: explanation and rule acquisition capabilities of the MYCIN system. Comput Biomed Res. 1975 Aug;8(4):303-20.

37. Miller PL. Critiquing anesthetic management: the "ATTENDING" computer system. Anesthesiology. 1983 Apr;58(4):362-9.

38. Miller PL. Extending computer-based critiquing to a new domain: ATTENDING, ESSENTIAL-ATTENDING, and VQ-ATTENDING. Int J Clin Monit Comput. 1986;2(3):135-42.

39. Masarie FE, Jr., Miller RA, Myers JD. INTERNIST-I properties: representing common sense and good medical practice in a computerized medical knowledge base. Comput Biomed Res. 1985 Oct;18(5):458-79.

40. Kuperman GJ, Gardner RM, Pryor TA. Help: a dynamic hospital information system. New York: Springer-Verlag; 1991.

41. Sittig DF. COMPAS: a computerized patient advice system to direct ventilatory care. [Dissertation]. Salt Lake City, UT: The University of Utah; 1988.

42. Sittig DF, Gardner RM, Pace NL, Morris AH, Beck E. Computerized management of patient care in a complex, controlled clinical trial in the intensive care unit. Comput Methods Programs Biomed. 1989 Oct-Nov;30(2-3):77-84.

43. Gardner RM, Golubjatnikov OK, Laub RM, Jacobson JT, Evans RS. Computer-critiqued blood ordering using the HELP system. Comput Biomed Res. 1990 Dec;23(6):514-28.

44. Evans RS, Pestotnik SL, Classen DC, et al. A computer-assisted management program for antibiotics and other antiinfective agents. N Engl J Med. 1998 Jan 22;338(4):232-8.

45. McDonald CJ. Protocol-based computer reminders, the quality of care and the non-perfectability of man. N Engl J Med. 1976 Dec 9;295(24):1351-5.

46. Geissbuhler A, Miller RA. Clinical application of the UMLS in a computerized order entry and decision-support system. Proc AMIA Symp. 1998:320-4.

47. Geissbuhler A, Miller RA. Distributing knowledge maintenance for clinical decision-support systems: the "knowledge library" model. Proc AMIA Symp. 1999:770-4.

48. Heusinkveld J, Geissbuhler A, Sheshelidze D, Miller R. A programmable rules engine to provide clinical decision support using HTML forms. Proc AMIA Symp. 1999:800-3.

49. Miller RA, Waitman LR, Chen S, Rosenbloom ST. The anatomy of decision support during inpatient care provider order entry (CPOE): empirical observations from a decade of CPOE experience at Vanderbilt. J Biomed Inform. 2005 Dec;38(6):469-85.

50. Waitman LR, Pearson D, Hargrove FR, et al. Enhancing Computerized Provider Order Entry (CPOE) for neonatal intensive care. AMIA Annu Symp Proc. 2003:1078.

51. Bates DW, Teich JM, Lee J, et al. The impact of computerized physician order entry on medication error prevention. J Am Med Inform Assoc. 1999 Jul-Aug;6(4):313-21.

52. Teich JM, Kuperman GJ, Bates DW. Clinical Decision Support: Making the Transition from the Hospital to the Community Network. Healthc Inf Manage. 1997;11(4):27-37.

53. Bates DW, Kuperman GJ, Rittenberg E, et al. A randomized trial of a computer-based intervention to reduce utilization of redundant laboratory tests. Am J Med. 1999 Feb;106(2):144-50.

54. Bates DW, Pappius E, Kuperman GJ, et al. Using information systems to measure and improve quality. Int J Med Inform. 1999 Feb-Mar;53(2-3):115-24.

55. Doolan DF, Bates DW, James BC. The use of computers for clinical care: a case series of advanced U.S. sites. J Am Med Inform Assoc. 2003 Jan-Feb;10(1):94-107.

56. Greenes RA, Sordo M, Zaccagnini D, Meyer M, Kuperman GJ. Design of a standards-based external rules engine for decision support in a variety of application contexts: report of a feasibility study at Partners HealthCare System. Medinfo. 2004;11(Pt 1):611-5.

57. Shojania KG, Yokoe D, Platt R, Fiskio J, Ma'luf N, Bates DW. Reducing vancomycin use utilizing a computer guideline: results of a randomized controlled trial. J Am Med Inform Assoc. 1998 Nov-Dec;5(6):554-62.

58. Payne TH, Hoey PJ, Nichol P, Lovis C. Preparation and use of preconstructed orders, order sets, and order menus in a computerized provider order entry system. J Am Med Inform Assoc. 2003 Jul-Aug;10(4):322-9.

59. Thompson TG, Brailer DJ. The Decade of Health Information Technology: Delivering Consumer-centric and Information-rich Health Care. Washington, DC: US Department of Health and Human Services; 2004.

60. Health Level 7. Arden Syntax for Medical Logic Systems Standard Version 2.5. Washington, DC: American National Standards Institute; 2004.

61. Hripcsak G. Arden Syntax for Medical Logic Modules. MD Comput. 1991 Mar-Apr;8(2):76, 8.

62. Hripcsak G, Ludemann P, Pryor TA, Wigertz OB, Clayton PD. Rationale for the Arden Syntax. Comput Biomed Res. 1994 Aug;27(4):291-324.

63. Jenders RA, Hripcsak G, Sideli RV, et al. Medical decision support: experience with implementing the Arden Syntax at the Columbia-Presbyterian Medical Center. Proc Annu Symp Comput Appl Med Care. 1995:169-73.

64. Karadimas HC, Chailloleau C, Hemery F, Simonnet J, Lepage E. Arden/J: an architecture for MLM execution on the Java platform. J Am Med Inform Assoc. 2002 Jul-Aug;9(4):359-68.

65. Maviglia SM, Zielstorff RD, Paterno M, Teich JM, Bates DW, Kuperman GJ. Automating complex guidelines for chronic disease: lessons learned. J Am Med Inform Assoc. 2003 Mar-Apr;10(2):154-65.

66. Ohno-Machado L, Gennari JH, Murphy SN, et al. The guideline interchange format: a model for representing guidelines. J Am Med Inform Assoc. 1998 Jul-Aug;5(4):357-72.

67. Patel VL, Allen VG, Arocha JF, Shortliffe EH. Representing clinical guidelines in GLIF: individual and collaborative expertise. J Am Med Inform Assoc. 1998 Sep-Oct;5(5):467-83.

68. Peleg M, Boxwala AA, Ogunyemi O, et al. GLIF3: the evolution of a guideline representation format. Proc AMIA Symp. 2000:645-9.

69. Wang D, Peleg M, Tu SW, et al. Design and implementation of the GLIF3 guideline execution engine. J Biomed Inform. 2004 Oct;37(5):305-18.

70. Peleg M, Boxwala AA, Tu S, et al. The InterMed approach to sharable computer-interpretable guidelines: a review. J Am Med Inform Assoc. 2004 Jan-Feb;11(1):1-10.

71. Sordo M, Boxwala AA, Ogunyemi O, Greenes RA. Description and status update on GELLO: a proposed standardized object-oriented expression language for clinical decision support. Medinfo. 2004;11(Pt 1):164-8.

72. Sordo M, Ogunyemi O, Boxwala AA, Greenes RA. GELLO: an object-oriented query and expression language for clinical decision support. AMIA Annu Symp Proc. 2003:1012.

73. Aho AV, Sethi R, Ullman JD. Compilers, principles, techniques, and tools. Reading, Mass.: Addison-Wesley Pub. Co.; 1986.

74. Grune D. Modern compiler design. Chichester; New York: Wiley; 2000.

75. OpenClinical. Summaries of guideline representation methods. 2006 [cited May 3, 2006]; Available from: http://www.openclinical.org/gmmsummaries.html

76. Parker CG, Rocha RA, Campbell JR, Tu SW, Huff SM. Detailed clinical models for sharable, executable guidelines. Medinfo. 2004;11(Pt 1):145-8.

77. Ram P, Berg D, Tu S, et al. Executing clinical practice guidelines using the SAGE execution engine. Medinfo. 2004;11(Pt 1):251-5.

78. Johnson PD, Tu SW, Musen MA, Purves I. A virtual medical record for guideline-based decision support. Proc AMIA Symp. 2001:294-8.

79. Kawamoto K, Lobach DF. Design, Implementation, Use, and Preliminary Evaluation of SEBASTIAN, a Standards-Based Web Service for Clinical Decision Support. Proc AMIA Symp. 2005.

80. Health Level 7. Patient Evaluation Service Draft Standard. Ann Arbor, MI; 2005.

81. HL7 Services Specification Project Workgroup, OMG Healthcare Domain Task Force. Healthcare Services Specification Project: The Business Case and Importance of Services (presentation). 2007 [cited 2007 2007 Feb 09]; Available from: http://hssp.wikispaces.com/space/showimage/2007-01_07_HSSP_Public_Slide_Deck_Version_1.2.ppt

82. Lamont J. Decision support systems prove vital to healthcare. KMWorld. 2007(01 Feb).

83. Glaser J, Flammini S. The Service-Oriented Solution for Health IT. HHN Most Wired. 2007(Feb 1).

84. Bakken S, Campbell KE, Cimino JJ, Huff SM, Hammond WE. Toward vocabulary domain specifications for health level 7-coded data elements. J Am Med Inform Assoc. 2000 Jul-Aug;7(4):333-42.

85. Dolin RH, Alschuler L, Beebe C, et al. The HL7 Clinical Document Architecture. J Am Med Inform Assoc. 2001 Nov-Dec;8(6):552-69.

86. Dolin RH, Alschuler L, Boyer S, et al. HL7 Clinical Document Architecture, Release 2. J Am Med Inform Assoc. 2006 Jan-Feb;13(1):30-9.

87. Huff SM. Clinical data exchange standards and vocabularies for messages. Proc AMIA Symp. 1998:62-7.

88. Spackman K. SNOMED RT and SNOMEDCT. Promise of an international clinical terminology. MD Comput. 2000 Nov-Dec;17(6):29.

89. Coonan KM. Medical informatics standards applicable to emergency department information systems: making sense of the jumble. Acad Emerg Med. 2004 Nov;11(11):1198-205.

90. Parrish F, Do N, Bouhaddou O, Warnekar P. Implementation of RxNorm as a Terminology Mediation Standard for Exchanging Pharmacy Medication between Federal Agencies. AMIA Annu Symp Proc. 2006:1057.

91. Gibson JT, Barker KN. Quality and comprehensiveness of the National Drug Code Directory on magnetic tape. Am J Hosp Pharm. 1988 Feb;45(2):337-40.

92. Cobb WM. The National Drug Code system: its purpose and potential. Hosp Formul. 1976 Nov;11(11):593-5.

93. US Department of Health and Human Services. Health Information Technology Initiative. Major Accomplishments: 2004-2006. 2007.

94. Halamka JD. Harmonizing healthcare data standards. J Healthc Inf Manag. 2006 Fall;20(4):11-3.

95. Diamond C, Ricciardi L. Building consumer trust into health information exchange. J Ahima. 2006 Nov-Dec;77(10):36, 8.

96. Markle Foundation. The Connecting for Health Common Framework: Resources for Implementing Private and Secure Health Information Exchange. New York; 2006.

97. Akamai Corporation. About Akamai. 2007 [cited 2007 Feb 09]; Available from: http://www.akamai.com/html/about/index.html

98. Corn M, Delaney C, Starren J. Grand Challenges for Informatics Research & Academics. Proc AMIA Spring Congress. 2006.

99. Osheroff JA, Teich JM, Middleton B, Steen EB, Wright A, Detmer DE. A Roadmap for National Action on Clinical Decision Support. J Am Med Inform Assoc. 2007 March-April;14(2):141-5.

100. Osheroff JA, Pifer EA, Teich JM, Sittig DF, Jenders RA. Improving Outcomes with Clinical Decision Support: An Implementer's Guide. Chicago, IL: HIMSS; 2005.

101. Berlin A, Sorani M, Sim I. A taxonomic description of computer-based clinical decision support systems. J Biomed Inform. 2006 Jan 9.

102. Wang JK, Shabot MM, Duncan RG, Polaschek JX, Jones DT. A clinical rules taxonomy for the implementation of a computerized physician order entry (CPOE) system. Proceedings / AMIA Annual Symposium. 2002:860-3.

103. Chaudhry B, Wang J, Wu S, et al. Systematic review: impact of health information technology on quality, efficiency, and costs of medical care. Annals of internal medicine. 2006 May 16;144(10):742-52.

104. Kuperman GJ, Teich JM, Gandhi TK, Bates DW. Patient safety and computerized medication ordering at Brigham and Women's Hospital. Jt Comm J Qual Improv. 2001 Oct;27(10):509-21.

105. Hongsermeier T, Kashyap V, Masson R. Collaborative Authoring of Decision Support Knowledge: A Demonstration. 2005 [cited 2006 Sep 5, 2006]; Available from: http://www.partners.org/cird/PPTS/CollabAuthDemo.ppt

106. Zielstorff RD, Teich JM, Paterno MD, et al. P-CAPE: a high-level tool for entering and processing clinical practice guidelines. Partners Computerized Algorithm and Editor. Proceedings / AMIA Annual Symposium. 1998:478-82.

107. Kuperman GJ, Teich JM, Bates DW, et al. Detecting alerts, notifying the physician, and offering action items: a comprehensive alerting system. Proc AMIA Annu Fall Symp. 1996:704-8.

108. Chertow GM, Lee J, Kuperman GJ, et al. Guided medication dosing for inpatients with renal insufficiency. Jama. 2001 Dec 12;286(22):2839-44.

109. Palchuk MB, Seger DL, Alexeyev A, et al. Implementing renal impairment and geriatric decision support in ambulatory e-prescribing. AMIA Annu Symp Proc. 2005:1071.

110. Palchuk M, Postilnik A, Vashevko M, et al. Smart Form Framework as a Foundation for Clinical Documentation Platform. Proceedings / AMIA Annual Symposium. 2006:(In press).

111. HL7. HL7 Electronic Health Record Functional Model and Standard. 2004.

112. Shafarman M, Van Hentenryck K. HL7 makes headway on Version 3: framers of the EHR draft standards invite the industry to try them out. Healthc Inform. 2004 Sep;21(9):50.

113. Sittig DF, Pace NL, Gardner RM, Beck E, Morris AH. Implementation of a computerized patient advice system using the HELP clinical information system. Computers and biomedical research, an international journal. 1989 Oct;22(5):474-87.

114. Classen DC, Avery AJ, Bates DW. Evaluation and certification of computerized provider order entry systems. J Am Med Inform Assoc. 2007 Jan-Feb;14(1):48-55.

115. Schadow G, Mead CN, Walker DM. The HL7 reference information model under scrutiny. Stud Health Technol Inform. 2006;124:151-6.

116. Smith B, Ceusters W. HL7 RIM: an incoherent standard. Stud Health Technol Inform. 2006;124:133-8.

117. Office of the National Coordinator for Health Information Technology. Presidential Initiatives: Consolidated Health Informatics. 2006 [cited 2007 Feb 15]; Available from: http://www.hhs.gov/healthit/chiinitiative.html

118. Calinforna Healthcare Foundation. ELINCS: Developing a National Lab Data Standard for EHRs. Oakland, CA: California Healthcare Foundation. 2005.

119. American Clinical Laboratory Association. ACLA Statement to the American Health Information Community on Laboratory Interfaces. 2006 [cited 2007 Feb 15]; Available from: http://www.clinical-labs.org/documents/AHIC103106statement.pdf

120. Forrey AW, McDonald CJ, DeMoor G, et al. Logical observation identifier names and codes (LOINC) database: a public use set of codes and names for electronic reporting of clinical laboratory test results. Clin Chem. 1996 Jan;42(1):81-90.

121. Huff SM, Rocha RA, McDonald CJ, et al. Development of the Logical Observation Identifier Names and Codes (LOINC) vocabulary. J Am Med Inform Assoc. 1998 May-Jun;5(3):276-92.

122. Regenstrief Institute. Logical Observation Identifiers Names and Codes (LOINC®). 2007 [cited 2007 Feb 15]; Available from: http://www.regenstrief.org/medinformatics/loinc/

123. Brouch K. AHIMA project offers insights into SNOMED, ICD-9-CM mapping process. J Ahima. 2003 Jul-Aug;74(7):52-5.

124. Matney S, Dent C, Rocha RA. Development of a compositional terminology model for nursing orders. Int J Med Inform. 2004 Aug;73(7-8):625-30.

125. Bowie J, Barnett GO. MUMPS--an economical and efficient time-sharing system for information management. Comput Programs Biomed. 1976 Apr;6(1):11-22.

126. Fitzpatrick B. Distributed caching with memcached. Linux Journal. 2004;2004(124).

127. Bates DW, Boyle DL, Vander Vliet MB, Schneider J, Leape L. Relationship between medication errors and adverse drug events. J Gen Intern Med. 1995 Apr;10(4):199-205.

128. Leape LL, Bates DW, Cullen DJ, et al. Systems analysis of adverse drug events. ADE Prevention Study Group. Jama. 1995 Jul 5;274(1):35-43.

129. Leape LL, Brennan TA, Laird N, et al. The nature of adverse events in hospitalized patients. Results of the Harvard Medical Practice Study II. N Engl J Med. 1991 Feb 7;324(6):377-84.

130. Stanaszek WF, Franklin CE. Survey of potential drug interaction incidence in an outpatient clinic population. Hosp Pharm. 1978 May;13(5):255-7, 61, 63.

131. Bates DW, Leape LL, Cullen DJ, et al. Effect of computerized physician order entry and a team intervention on prevention of serious medication errors. Jama. 1998 Oct 21;280(15):1311-6.

132. Bates DW, O'Neil AC, Boyle D, et al. Potential identifiability and preventability of adverse events using information systems. J Am Med Inform Assoc. 1994 Sep-Oct;1(5):404-11.

133. Feldstein AC, Smith DH, Perrin N, et al. Reducing warfarin medication interactions: an interrupted time series evaluation. Arch Intern Med. 2006 May 8;166(9):1009-15.

134. Kuperman GJ, Bobb A, Payne TH, et al. Medication-related clinical decision support in computerized provider order entry systems: a review. J Am Med Inform Assoc. 2007 Jan-Feb;14(1):29-40.

135. Neuvonen PJ, Penttila O. Effect of oral ferrous sulphate on the half-life of doxycycline in man. Eur J Clin Pharmacol. 1974 Aug 23;7(5):361-3.

136. Spina JR, Glassman PA, Belperio P, Cader R, Asch S. Clinical relevance of automated drug alerts from the perspective of medical providers. Am J Med Qual. 2005 Jan-Feb;20(1):7-14.

137. van der Sijs H, Aarts J, Vulto A, Berg M. Overriding of drug safety alerts in computerized physician order entry. J Am Med Inform Assoc. 2006 Mar-Apr;13(2):138-47.

138. Weingart SN, Toth M, Sands DZ, Aronson MD, Davis RB, Phillips RS. Physicians' decisions to override computerized drug alerts in primary care. Arch Intern Med. 2003 Nov 24;163(21):2625-31.

139. Greim JA, Shek C, Jones L, et al. Enterprise-wide drug-drug interaction alerting system. AMIA Annu Symp Proc. 2003:856.

140. Shah NR, Seger AC, Seger DL, et al. Improving acceptance of computerized prescribing alerts in ambulatory care. J Am Med Inform Assoc. 2006 Jan-Feb;13(1):5-11.

141. Bavdekar SB, Pawar M. Evaluation of an Internet delivered pediatric diagnosis support system (ISABEL) in a tertiary care center in India. Indian Pediatr. 2005 Nov;42(11):1086-91.

142. Ramnarayan P, Tomlinson A, Rao A, Coren M, Winrow A, Britto J. ISABEL: a web-based differential diagnostic aid for paediatrics: results from an initial performance evaluation. Arch Dis Child. 2003 May;88(5):408-13.

143. Ramnarayan P, Winrow A, Coren M, et al. Diagnostic omission errors in acute paediatric practice: impact of a reminder system on decision-making. BMC Med Inform Decis Mak. 2006;6:37.

144. Ramnarayan P, Roberts GC, Coren M, et al. Assessment of the potential impact of a reminder system on the reduction of diagnostic errors: a quasi-experimental study. BMC Med Inform Decis Mak. 2006;6:22.

145. Fick DM, Cooper JW, Wade WE, Waller JL, Maclean JR, Beers MH. Updating the Beers criteria for potentially inappropriate medication use in older adults: results of a US consensus panel of experts. Arch Intern Med. 2003 Dec 8-22;163(22):2716-24.

146. van der Hooft CS, Jong GW, Dieleman JP, et al. Inappropriate drug prescribing in older adults: the updated 2002 Beers criteria--a population-based cohort study. Br J Clin Pharmacol. 2005 Aug;60(2):137-44.

147. Benkler Y. Coase's Penguin, or, Linux and the Nature of the Firm. Yale Law Journal. 2002;112(3):367-445.

148. Leuf B, Cunningham W. The Wiki way: quick collaboration on the Web: Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA; 2001.

149. Gorman PN, Helfand M. Information seeking in primary care: how physicians choose which clinical questions to pursue and which to leave unanswered. Med Decis Making. 1995 Apr-Jun;15(2):113-9.

150. Covell DG, Uman GC, Manning PR. Information needs in office practice: are they being met? Ann Intern Med. 1985 Oct;103(4):596-9.

151. Ely JW, Burch RJ, Vinson DC. The information needs of family physicians: case-specific clinical questions. J Fam Pract. 1992 Sep;35(3):265-9.

152. Westbrook JI, Gosling AS, Coiera E. Do clinicians use online evidence to support patient care? A study of 55,000 clinicians. J Am Med Inform Assoc. 2004 Mar-Apr;11(2):113-20.

153. Schilling LM, Steiner JF, Lundahl K, Anderson RJ. Residents' patient-specific clinical questions: opportunities for evidence-based learning. Acad Med. 2005 Jan;80(1):51-6.

154. Campbell R, Ash J. Comparing bedside information tools: a user-centered, task-oriented approach. AMIA Annu Symp Proc. 2005:101-5.

155. Cimino JJ, Li J, Graham M, et al. Use of online resources while using a clinical information system. AMIA Annu Symp Proc. 2003:175-9.

156. Maviglia SM, Yoon CS, Bates DW, Kuperman G. KnowledgeLink: impact of context-sensitive information retrieval on clinicians' information needs. J Am Med Inform Assoc. 2006 Jan-Feb;13(1):67-73.

157. Columbia University Center for Advanced Information Management. Info Button Manager Fact Sheet. 2006 [cited 2007 19 Feb]; Available from: http://www.cat.columbia.edu/pdfs/Info_Button_Mgr_2006.pdf

158. National Research Council (U.S.). Committee on Academic Careers for Experimental Computer Scientists. Academic careers for experimental computer scientists and engineers. Washington, DC: National Academy Press; 1994.

159. Stead WW, Haynes RB, Fuller S, et al. Designing medical informatics research and library--resource projects to increase what is learned. J Am Med Inform Assoc. 1994 Jan-Feb;1(1):28-33.

160. Sosin DM, DeThomasis J. Evaluation challenges for syndromic surveillance--making incremental progress. MMWR Morb Mortal Wkly Rep. 2004 Sep 24;53 Suppl:125-9.

161. Townes JM, Kohn MA, Southwick KL, et al. Investigation of an electronic emergency department information system as a data source for respiratory syndrome surveillance. J Public Health Manag Pract. 2004 Jul-Aug;10(4):299-307.

162. Bravata DM, McDonald KM, Smith WM, et al. Systematic review: surveillance systems for early detection of bioterrorism-related diseases. Ann Intern Med. 2004 Jun 1;140(11):910-22.

163. Lombardo J, Burkom H, Elbert E, et al. A systems overview of the Electronic Surveillance System for the Early Notification of Community-Based Epidemics (ESSENCE II). J Urban Health. 2003 Jun;80(2 Suppl 1):i32-42.

164. Muscatello DJ, Churches T, Kaldor J, et al. An automated, broad-based, near real-time public health surveillance system using presentations to hospital Emergency Departments in New South Wales, Australia. BMC Public Health. 2005;5:141.

165. Platt R, Bocchino C, Caldwell B, et al. Syndromic surveillance using minimum transfer of identifiable data: the example of the National Bioterrorism Syndromic Surveillance Demonstration Program. J Urban Health. 2003 Jun;80(2 Suppl 1):i25-31.

166. Espino JU, Wagner M, Szczepaniak C, et al. Removing a barrier to computer-based outbreak and disease surveillance--the RODS Open Source Project. MMWR Morb Mortal Wkly Rep. 2004 Sep 24;53 Suppl:32-9.

167. Wagner MM, Espino J, Tsui FC, et al. Syndrome and outbreak detection using chief-complaint data--experience of the Real-Time Outbreak and Disease Surveillance project. MMWR Morb Mortal Wkly Rep. 2004 Sep 24;53 Suppl:28-31.

168. Tierney WM, McDonald CJ, Martin DK, Rogers MP. Computerized display of past test results. Effect on outpatient testing. Ann Intern Med. 1987 Oct;107(4):569-74.

169. Kingsland LC, 3rd, Lindberg DA, Sharp GC. AI/RHEUM. A consultant system for rheumatology. J Med Syst. 1983 Jun;7(3):221-7.

170. Sittig DF. Grand challenges in medical informatics? J Am Med Inform Assoc. 1994 Sep-Oct;1(5):412-3.

171. Berner ES. Clinical decision support systems: theory and practice. New York: Springer; 1999.

172. Jimison H, Sher P. Clinical decision support systems: theory and practice. In: Berner ES, editor. Health informatics. New York: Springer; 1999. p. 139-66.

173. Buckland A. Dr. Weed's computer world. J Am Med Rec Assoc. 1987 Jul;58(7):21-3.

174. Miller RA, Schaffner KF, Meisel A. Ethical and legal issues related to the use of computer programs in clinical medicine. Ann Intern Med. 1985 Apr;102(4):529-37.

175. McCord G, Smucker WD, Selius BA, et al. Answering questions at the point of care: do residents practice EBM or manage information sources? Acad Med. 2007 Mar;82(3):298-303.

176. Fenton SH, Badgett R. Are there differences in online resources for answering primary care questions? AMIA Annu Symp Proc. 2005:953.

177. Leff B, Harper GM. The reading habits of medicine clerks at one medical school: frequency, usefulness, and difficulties. Acad Med. 2006 May;81(5):489-94.

178. Koonce TY, Giuse NB, Todd P. Evidence-based databases versus primary medical literature: an in-house investigation on their optimal use. J Med Libr Assoc. 2004 Oct;92(4):407-11.

179. Peterson MW, Rowat J, Kreiter C, Mandel J. Medical students' use of information resources: is the digital age dawning? Acad Med. 2004 Jan;79(1):89-95.

180. Iakovidis I. Towards personal health record: current situation, obstacles and trends in implementation of electronic healthcare record in Europe. Int J Med Inf. 1998;52(1-3):105-15.

181. Kim MI, Johnson KB. Personal Health Records: Evaluation of Functionality and Utility. Journal of the American Medical Informatics Association. 2002;9(2):171.

182. Sittig DF. Personal health records on the internet: a snapshot of the pioneers at the end of the 20th Century. Int J Med Inf. 2002;65(1):1-6.

183. Denton IC. Will Patients Use Electronic Personal Health Records? Responses from a Real-Life Experience. Journal of Healthcare Information Management. 2001;15(3):251-9.

184. Tang PC, Ash JS, Bates DW, Overhage JM, Sands DZ. Personal health records: definitions, benefits, and strategies for overcoming barriers to adoption. J Am Med Inform Assoc. 2006 Mar-Apr;13(2):121-6.

185. Doherty WJ, Thadhani AJ. The economic value of rapid response time. IBM Report.

186. Hamann C, Poon E, Smith S, et al. Designing an electronic medication reconciliation system. AMIA Annu Symp Proc. 2005:976.

187. Poon EG, Blumenfeld B, Hamann C, et al. Design and implementation of an application and associated services to support interdisciplinary medication reconciliation efforts at an integrated healthcare delivery network. J Am Med Inform Assoc. 2006 Nov-Dec;13(6):581-92.

188. Gorman P. RxSafe: Using IT to Improve Medication Safety for Rural Elders. 2006 [cited 2007 Feb 09]; Available from: http://medir.ohsu.edu/~gormanp/slides/051003_RxSafe2_HO.pdf

189. Campbell JR, Payne TH. A comparison of four schemes for codification of problem lists. Proc Annu Symp Comput Appl Med Care. 1994:201-5.

190. Gross PA, Bates DW. A pragmatic approach to implementing best practices for clinical decision support systems in computerized provider order entry systems. J Am Med Inform Assoc. 2007 Jan-Feb;14(1):25-8.

191. Gao T, Ma H, Yen IL, Bastani F, Tsai WT. Toward QoS Analysis of Adaptive Service-Oriented Architecture. Service-Oriented System Engineering, 2005 SOSE 2005 IEEE International Workshop. 2005:227-36.

192. Kohlhoff C, Steele R. Evaluating SOAP for High Performance Business Applications: Real-Time Trading Systems. Proceedings of WWW2003. 2003:03-2002.

193. Liu H, Lin X, Li M. Modeling response time of SOAP over Http. Web Services, 2005 ICWS 2005 Proceedings 2005 IEEE International Conference on. 2005:679.

194. Papazoglou MP. Service-oriented computing: concepts, characteristics and directions. Web Information Systems Engineering, 2003 WISE 2003 Proceedings of the Fourth International Conference on. 2003:3-12.

195. Seshasayee B, Schwan K, Widener P. SOAP-binQ: high-performance SOAP with continuous quality management. Distributed Computing Systems, 2004 Proceedings 24th International Conference on. 2004:158-65.

196. Sharma A, Adarkar H, Sengupta S, Limited IT. Managing QoS through prioritization in Web services. Web Information Systems Engineering Workshops, 2003 Proceedings Fourth International Conference on. 2003:140-8.

197. W3C Web Services Architecture Working Group. QoS for Web Services: Requirements and Possible Approaches. Boston, MA: W3C; 2003.

198. Wang G, Chen A, Wang C, Fung C, Uczekaj S. Integrated quality of service (QoS) management in service-oriented enterprise architectures. Enterprise Distributed Object Computing Conference, 2004 EDOC 2004 Proceedings Eighth IEEE International. 2004:21-32.