An Interoperable Framework for a Clinical Decision Support System

by

Iryna Bilykh

B.S.B.A., Central Missouri State University, 2001

A Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

...ized standard

Supervisor: Dr. Jens H. Jahnke

Abstract

The healthcare sector is facing a significant challenge: delivering quality clinical care in a costly and intricate environment. There is a general consensus that a solution for many aspects of this problem lies in establishing a framework for effective and efficient clinical decision support.

The key to good decision support is offering clinicians just-in-time accessibility to relevant patient specific knowledge. However, at the present time, management of clinical knowledge and patient records is significantly inadequate resulting in sometimes uninformed, erroneous, and costly clinical decisions.

One of the contributing factors is that the field of healthcare is characterized by large volumes of highly complex medical knowledge and patient information that must be captured, processed, interpreted, stored, analyzed, and exchanged. Moreover, different clinical information systems are typically not interoperable.

This thesis introduces an approach for realizing a clinical decision support framework that manages complex clinical knowledge in a form of evidence-based clinical practice guidelines. The focus of presented work is directed on the interoperability of knowledge, information, and processes in a heterogeneous distributed environment.

The main contributions of this thesis include definition of requirements, conceptual architecture, and approach for an interoperable clinical decision support system that is stand-alone, independent, and based on open source standards.

Examiners:

Table of Contents

ABSTRACT ..... **II**

TABLE OF CONTENTS ..... **IV**

CHAPTER 1 INTRODUCTION **1**

1.1 MOTIVATION 1

1.2 CHALLENGES FOR INTEROPERABILITY 2

1.3 THESIS OBJECTIVES 4

1.4 KEY CONTRIBUTIONS 5

1.5 THESIS OUTLINE 6

CHAPTER 2 RELATED WORK **8**

2.1 KNOWLEDGE REPRESENTATION LANGUAGES 8

2.1.1 Arden Syntax *9*
2.1.2 Guideline Interchange Format *13*
2.1.3 PROforma *15*
2.2 CLINICAL INFORMATION STANDARDS 17

2.2.1 Health Level 7 ..... *17*
2.2.1.1 Reference Information Model *18*
2.2.1.2 Clinical Document Architecture *18*
2.2.2 Medical Terminologies *19*
2.2.2.1 Unified Medical Language System *20*
2.3 CLINICAL DECISION SUPPORT SYSTEMS 21

2.3.1 Leeds *21*
2.3.2 MYCIN *22*
2.3.3 HELP *23*
2.3.4 EON *23*
2.4 REASONING METHODOLOGIES FOR DECISION SUPPORT 24

2.4.1 Symbolic Reasoning *25*
2.4.1.1 Rule-based Model *25*
2.4.1.2 Case-based Model *27*
2.4.2 Sub-Symbolic Reasoning *29*
2.4.2.1 Artificial Neural Networks *29*
2.4.2.2 Bayesian Networks *31*
2.5 INTEROPERABILITY OF DISTRIBUTED SERVICE-ORIENTED SYSTEMS 32

2.5.1 Technologies for Service-Oriented Architectures *33*
2.5.1.1 CORBA *33*
2.5.1.2 CORBA Services *35*
2.5.1.2.1 Naming Service *36*
2.5.1.2.2 Trading Object Service *36*
2.5.1.2.3 Transaction Service *36*

2.5.1.2.4 Security Service ... 37
2.5.1.3 Web Services ... 37
2.5.1.4 Web Service Stack ... 38
2.5.1.4.1 Transport ... 38
2.5.1.4.2 Messaging ... 39
2.5.1.4.3 Secure Messaging ... 39
2.5.1.4.4 Service Description ... 40
2.5.1.4.5 Service Discovery ... 40
2.5.1.4.6 Service Flow ... 41
2.5.1.4.7 Semantics ... 42

CHAPTER 3 REQUIREMENTS ... **43**

3.1 KNOWLEDGE INTEROPERABILITY REQUIREMENTS ... 43

3.1.1 Open Content License ... 43
3.1.2 Formal Semantics ... 43
3.1.3 Formal Syntax ... 44
3.1.4 Expressive Decision Model ... 45
3.1.5 Integration with a Standard Patient Information Model ... 47
3.1.6 Integration with Standard Medical Ontologies ... 47
3.1.7 Conformance to Other Existing Standards ... 47
3.1.8 Broadly Known Formalism ... 48
3.1.9 Structured Recommendations ... 48

3.2 INFORMATION INTEROPERABILITY REQUIREMENTS ... 48

3.2.1 Open Content License ... 49
3.2.2 Standard Status ... 49
3.2.3 Defined Structure ... 49
3.2.4 Content Specification Process ... 50
3.2.5 Compatibility with Existing Tools ... 50

3.3 PROCESS INTEROPERABILITY REQUIREMENTS ... 51

3.3.1 Platform-Agnostic Middleware ... 51
3.3.2 Transaction Management ... 51
3.3.3 Document-Oriented Messaging ... 52
3.3.4 Communication Protocol ... 52

CHAPTER 4 APPROACH ... **54**

4.1 KNOWLEDGE INTEROPERABILITY APPROACH ... 54

4.1.1 Comparison Matrix ... 54
4.1.2 Arden Syntax ... 55
4.1.2.1 Formal Semantics ... 56
4.1.2.2 Integration with a Standard Information Model ... 56
4.1.2.3 Integration with Standard Medical Ontologies ... 58
4.1.2.4 Structured Recommendations ... 58

4.2 INFORMATION INTEROPERABILITY APPROACH ... 59

4.2.1 Clinical Document Architecture ... 59
4.2.1.1 Defined Structure ... 60
4.2.1.2 Content Specification Process ... 62
4.2.1.3 W3C Schema Restriction ... 63
4.2.2 CDA Level Two Templates ... 64
4.2.2.1 Core Data Set ... 64
4.2.2.2 Additional Data Request ... 66
4.2.2.3 Extended Data Set ... 66

4.2.2.4 Recommendations *67*
4.2.3 Leveraging Existing Tools *67*
4.3 PROCESS INTEROPERABILITY APPROACH 68

4.3.1 Middleware *69*
4.3.1.1 Scalability *69*
4.3.1.2 Firewall Traversal *69*
4.3.1.3 Vendor Independence *70*
4.3.1.4 Leveraging the Web *71*
4.3.1.5 Transaction Management *71*
4.3.1.6 Communication Protocol *72*

CHAPTER 5 CONCEPTUAL ARCHITECTURE **74**

5.1 COMPONENT ARCHITECTURE 74

5.1.1 Web Server *74*
5.1.2 Transaction Manager *75*
5.1.3 Document Manager *76*
5.1.4 Schema Repository *77*
5.1.5 Medical Ontology Server *77*
5.1.6 Guideline Repository *78*
5.1.7 Guideline Preprocessor *78*
5.1.8 Reasoning Engine *79*
5.2 COMPONENT INTERFACE SPECIFICATIONS 79

CHAPTER 6 DISCUSSION **84**

6.1 KNOWLEDGE INTEROPERABILITY 84

6.1.1 Arden Syntax and Formal Semantics *84*
6.1.2 Curly Braces Problem *85*
6.1.3 Arden Syntax and Complex Guidelines *86*
6.2 INFORMATION INTEROPERABILITY 87

6.2.1 Clinical Document Architecture *87*
6.3 PROCESS INTEROPERABILITY 89

6.3.1 Communication Protocol *89*

CHAPTER 7 CONCLUSION **91**

7.1 SUMMARY 91

7.2 FUTURE WORK 92

List of Tables

TABLE 1: GUIDELINE LANGUAGES COMPARISON 54

List of Figures

FIGURE 1: ARDEN SYNTAX GUIDELINE 11
FIGURE 2: CURLY BRACES PROBLEM ILLUSTRATION 12
FIGURE 3: GLIF GUIDELINE 14
FIGURE 4: PROFORMA GUIDELINE 16
FIGURE 5: CLINICAL RULE EXAMPLE 26
FIGURE 6: CORBA MODEL 35
FIGURE 7: WEB SERVICES STACK 38
FIGURE 8: CURLY BRACES SOLUTION EXAMPLE 57
FIGURE 9: CDA LEVEL TWO DOCUMENT STRUCTURE 62
FIGURE 10: CDSS COMMUNICATION PROTOCOL 73
FIGURE 11: CONCEPTUAL COMPONENT ARCHITECTURE 74
FIGURE 12: COMPONENT INTERFACE DEPENDENCIES 80

Acknowledgements

On a personal note, I would like to express my deepest gratitude to my supervisor, Dr. Jens Jahnke for his guidance and support. Special thank you to Dr. Morgan Price for granting continuous encouragement, mentorship, and sharing clinical knowledge.

I would also like to acknowledge my parents who made my academic pursuits possible.

To Inda

for being my best friend

Chapter 1 Introduction

1.1 Motivation

The healthcare sector is facing a significant challenge: delivering quality clinical care in a costly and intricate environment. Canada, being the third most competitive economy in the world [1], spends more on healthcare than the majority of other countries. In the year 2001, nearly 10% of the Gross Domestic Product (GDP) went on healthcare, up from 7.3% in 1981 [2]. These numbers indicate that reducing the costs of healthcare by improving its quality is critical to the future state of our society.

There is a general consensus that a solution for many aspects of this problem lies in establishing a framework for effective and efficient clinical decision support [3]. It has been shown that computer-based decision support has a positive effect on physician performance, patient outcomes [4], and medical error rate reduction [5].

The key to good decision support is offering clinicians just-in-time accessibility to relevant patient specific knowledge.

At the present time, management of clinical knowledge and patient records is significantly inadequate resulting in sometimes uninformed, erroneous, and costly clinical decisions [6]. In fact, healthcare informatics domain is notably behind in a knowledge management sense, if compared to other industries such as automobile, airline, or banking sectors.

One of the contributing factors is that the field of healthcare is characterized by large volumes of highly complex data that must be captured, processed, interpreted, stored, analyzed, and exchanged. The difficulty of dealing with such information is a major

obstacle for using informatics to support health care and fulfill strategic goals. Moreover, different clinical information systems are not built with interoperability "in mind" and thus employ heterogeneous technologies on inter- and intra-organizational levels. Therefore, the state-of-the-art in the clinical informatics domain poses a number of challenges for realizing an interoperable decision support solution [7]. It must be stated, meanwhile, that the work presented in this thesis focuses primarily on interoperability issues of Clinical Decision Support Systems (CDSS). Aspects of knowledge processing and automated reasoning are addressed in an upcoming thesis by Glen McCallum[1].

1.2 Challenges for Interoperability

The key challenges of information sharing for effective and efficient clinical decision support are:

- **Knowledge interoperability**

Here we define knowledge as a part of three-tier hierarchy: data, information, and knowledge. Data represents raw facts; information is data in a specific context; knowledge is information with purpose and guidance based on evidence, insight, and experience [8]. In the medical community, clinical knowledge comes from multiple sources and is expressed in various forms. Currently the majority of clinical knowledge such as literature, references, and guidelines is text-based and represented as unstructured narratives [9]. Therefore, vast amount of knowledge is not easily accessible to the clinicians at the point of care, when it is most needed. Even electronic representation of knowledge is often in the inadequate format and time constraints prohibit the clinicians to search and interpret it manually. The burdens of information overload become

[1] Netlab group, Department of Computer Science, University of Victoria.

significant obstacles in knowledge accessibility and clinical decision making [10]. Therefore, in order to support the information needs of clinicians and ensure quality of care delivery, adequate knowledge processing and decision support tools are required.

- **Information interoperability**

Nowadays a patient is rarely treated by a single clinician within one organization. Typically a patient is handled by different health care units that provide specialized services such as laboratory, pharmacy, physiotherapy, surgery, etc. Often these health care units are spread over inter-organizational and regional levels. Such state of patient management results in patient information being stored in multiple sources that are often isolated and not easily accessible.

Therefore, an interoperable information framework is necessary for allowing health care professionals to securely access accurate and timely patient information from distributed systems. Patient information availability through integration can significantly improve healthcare delivery and reduce its costs by eliminating the duplication of expensive services as well as improving clinical decision making [11].

- **Process interoperability**

Process interoperability refers to the ability of a CDSS to be integrated directly into clinical workflows. As we discuss further in this thesis (see Section 2.3), there exists a number of CDSSs that achieve process interoperability by being tightly coupled with specific Electronic Medical Record (EMR) systems. Such approach is not suitable for a CDSS that must be independent of any particular EMR. Rather than "hard-wiring" its functionality into the EMR, a CDSS should define its process interoperability constructs as system-agnostic transaction protocols on the interface level. These transactions should

handle the extraction of patient information from an EMR and providing clinical knowledge with recommendations in return. Additionally, the technical aspect of process interoperability encompasses implementation platform, programming language, and location transparencies. However, to the present day a consensus on common interoperability standards for clinical systems is lacking while existing systems vary significantly from one organization to another.

Thus, adequate technical solutions are needed for designing a CDSS that can interoperate potentially with any EMR by providing system-agnostic process interoperability and taking an active role in the definition and execution of transactions.

1.3 Thesis Objectives

Our presented work addresses the above mentioned problems of interoperability in the clinical decision support domain. The two main objectives of this thesis are listed below:

1. Our main objective is to define a standard-compliant CDSS framework that can interoperate with a variety of clinical systems and provide clinicians with patient-specific knowledge in the form of evidence-based clinical practice guidelines. Clinical guidelines are defined as sets of schematic plans, at varying levels of abstraction and detail, for management of patients who have a particular clinical condition [12]. Guidelines can represent knowledge in a structured form and, therefore, are suitable constructs for encoding knowledge on a computer-interpretable level. The work in this thesis primarily addresses the preventive care guidelines as they are significantly less complex than diagnosis guidelines or advanced care plans. There are many aspects involved in guideline-based clinical decision support, among which are guideline authoring and maintenance, retrieval, and run-time execution.

However, the intention of this thesis is to define a CDSS framework that primarily focuses on the interoperability functionalities.

2. Another goal of this work is to contribute our findings to the academic and industrial community of CDSS implementers. The work presented in this thesis is a synthesis of ideas generated by a group of experienced clinicians, informaticians, and software engineers who embarked on a collaborative effort to design an interoperable CDSS framework.

1.4 Key Contributions

During the research of related work we have found that there are already a number of various products designed to provide the functionality of clinical decision support (see Section 2.3). However, the majority of CDSSs can be described by one or more of the following characteristics: tightly imbedded into specific EMRs, proprietary, or focused on narrow clinical problem domains. In addition, there is a lack of consistency in standards related to interoperability, data sharing, and knowledge representation. In comparison with the existing CDSSs, our proposed CDSS offers the following contributions:

- Interoperability is indiscriminative towards EMRs as the system is based on technical and data standards (see Section 4.2).
- Abstraction of the CDSS interoperability layer from the reasoning modules allows independent evolution of the CDSS components. This separation of communication and reasoning concerns is beneficial for the systems where continuous system change and improvement is anticipated.

- Mediation of knowledge, patient information, and process flows on an intermediary layer facilitates a gateway between an EMR and potentially any number of specialized CDSS implementations. This approach relieves the EMR from the burden of awareness and management of multiple CDSSs.

1.5 Thesis Outline

To meet the objectives outlined in Section 1.3, the thesis is structured as follows. Chapter 2 introduces related work, including relevant technologies and projects. The analysis of the CDSS domain provides a direction for our work as well as illustrates a summary of our investigations and studies in the health informatics field.

In Chapter 3 we present the requirements identified for the interoperable CDSS framework. The requirements were collected through a process of iterative discussions with software engineers and clinical domain experts.

Chapter 4 addresses our approach to knowledge, information, and process interoperability. Here we specifically focus on clinical practice guidelines as constructs for knowledge representation and sharing, patient information models, as well as the details of process interoperability and middleware

The architecture for the proposed CDSS is outlined in Chapter 5 which also includes a detailed description of identified system components. The purpose of this chapter is to provide a roadmap for designing a CDSS. The framework itself is not targeted for a particular prototype, but rather toward a product line of interoperable stand-alone decision support systems.

As a discussion of the overall research work, major challenges associated with realizing and designing a CDSS are presented in Chapter 6 followed by the concluding remarks in Chapter 7.

Chapter 2 Related Work

This chapter provides a detailed examination of relevant work pertinent to automated clinical decision support. Here we discuss several major guideline encoding languages as well as information interoperability standards including structured medical terminologies. Furthermore, we look at several clinical decision support systems that were successfully implemented in the academic or clinical settings. Finally, an overview of various computer-based reasoning methodologies for decision making is presented followed by a discussion of technologies for realizing interoperable distributed architectures.

2.1 Knowledge Representation Languages

As a medium for structured knowledge encoding we have chosen clinical practice guidelines. It is important to note that we have also limited our scope to addressing only preventive care guidelines.

Guidelines can be described as "Systematically developed statements to assist practitioners and patient decisions about appropriate health care for specific clinical circumstances" [13]. Nowadays, guidelines are developed by health authorities and are typically distributed to the health care providers in many (often paper-based) formats: narrative text, decision tables, graphs, flowcharts, if-then statements, etc.

However, automation of guideline-based decision support requires the use of a structured guideline encoding language. Such language should be richly expressive, machine readable, and able to specify various types of clinical actions, temporal and other constraints, as well as outcome intentions of the guideline [12]. The purpose of this section is to review several most developed guideline encoding languages that satisfy

these criteria and specifically are appropriate for preventive care guidelines. More information on comparison of these and other guideline representation models can be found in [14] and in [15].

2.1.1 Arden Syntax

Arden syntax is a procedural rule-based formalism for sharing computerized clinical knowledge [16]. It is non-proprietary, modular, and system independent language designed for supporting clinical decision making.

Arden is the longest established guideline representation language. It was first introduced in 1989 and its version 2.0 was adopted by HL7 in 1999. Moreover, it became one of American National Standard Institute (ANSI) standards in year 2002. Because this guideline representation language has been around for a significant length of time and is currently under active development, there is a large body of academic and practical work around it. Arden differs from other guideline representation languages such as GLIF and PROforma in the sense that it is the only standard for procedural representation of clinical knowledge.

Arden represents medical algorithms containing condition-action rules as Medical Logic Modules (MLMs). An MLM is the main construct of Arden syntax. MLMs are stored as ASCII text files and can be created with virtually any text editor. In addition to decision logic, an MLM can contain information for managing MLM knowledge base; as well as links to other MLMs or external knowledge sources. Some examples of MLM types can be recommendations, alerts, data interpretations, diagnostic scores, etc. An MLM itself represents an independent unit of logic for a single decision step, thus in

order to encode a multi-step clinical practice guideline, a series of MLMs have to be combined.

The main advantage of using Arden is its simplicity because it does not include complex structures and does not require extensive programming experience for building and interpreting MLMs. In fact, Arden MLMs are intended to be created and used by the clinicians. Moreover, Arden syntax is formalized in a context free grammar, specifically Backus-Naur Form (BNF) which is a critical characteristic for a guideline language intended for automated decision support. Another advantageous characteristic of Arden syntax is support for time functions which has high relevance for integration of MLMs with clinical events. Moreover, Arden also provides clear and explicit "hooks" for embedding MLMs into local clinical information systems.

The following example MLM [17] contains decision logic for monitoring patients with elevated potassium level (hyperkalemia). This MLM is used to demonstrate Arden syntax.

```
maintenance:
    title: Screen for hyperkalemia in critical value range (>6.0);;
    filename: HYPERKALEMIA;;
    version: 1;;
    institution: Columbia-Presbyterian Medical Center;;
    author: Pete Stetson (peter.stetson@dbmi.columbia.edu);;
    specialist: Jai Radhakrishnon, MD, John Crew, MD;;
    date: 2003-09-16;;
    validation: test;;
library:
    purpose: To monitor for patients who have a critically elevated potassium level;;
    explanation: When a potassium lab result is stored, a warning is sent if it is >6.0
    mg/dl. If the patient is in renal failure a lower threshold K+ valued is used;;
    keywords: potassium, hyperkalemia;;
knowledge:
    type: data-driven;;
    data:
        ...
        raw_potassiums := read last 3 from {'dam'="PDQRES2";;
                            '1301','1608','1609','1610','1656',1698','32713','33803', '35455',
                            '35975','35993','35994'} where they occurred within the past 3 months);
        ...
        ;;
    evoke: k_storage_event;;
    logic:
        ...
        creatinine :=last(raw_creatinine where it is number);
        ...
        if potassium >= cut_off then
            conclude true;
        else
            conclude false;
        endif;
        ...
        ;;
    action:
        write "This patient has a critically elevated K+ of " || potassium ||
        ..."
        ;;
    end:
```

Figure 1: Arden Syntax guideline

Arden does have strong potential for facilitating knowledge sharing among clinical information system; however interoperability can easily be hindered by the tight coupling of MLM data elements to the local data models. As a consequence, an MLM developed for a particular clinical system would be difficult to port to another system. Tight coupling occurs because in Arden local data elements are referenced within curly braces that contain queries to the local data [18]. Unfortunately, there is no specification neither for content or structure for what can be placed inside of the curly braces. This is the major limitation of Arden and is often referred to as "curly braces problem". This impediment can be illustrated by the following excerpt from the above example MLM where the last three potassium test result are retrieved from the EMR:

```
raw_potassiums := read last 3 from {'dam'="PDQRES2";;
'1301','1608','1609','1610','1656',1698','32713','33803', '35455', '35975','35993','35994'} where
they occurred within the past 3 months);
```

Figure 2: Curly braces problem illustration

The query enclosed in the curly braces represents the EMR specific data mapping that is not part of the Arden standard. Such site specificity therefore can make a particular MLM unfit for sharing with other EMR systems that have other internal database structures. The future versions of Arden are actively worked on by HL7 and may address this limitation of the "curly braces problem" by defining a standard query model. Nonetheless, because of its relative simplicity, Arden has a high adoption rate in many health information systems and many MLMs exist. The major vendors of clinical systems that have implemented Arden include Eclipsys, McKesson, IBM, Siemens, and Cerner. Unfortunately, most Arden implementation tools are developed by commercial

vendors and are not available outside their proprietary systems. This is another drawback for adopting Arden.

2.1.2 Guideline Interchange Format

Guideline Interchange Format (GLIF) is a computer-interpretable language for guideline modeling and execution [19]. GLIF encompasses ontology for representing guideline constructs, as well as ontology of medical concepts. It was developed by InterMed Collaboratory consisting of Stanford, Harvard, McGill, and Columbia universities. In the year 2000 the most recent specification of GLIF was released. Early versions of GLIF were based on a proprietary Object Data Interchange Format (ODIF) based syntax [20], which is now replaced in GLIF by XML.

GLIF is designed to support guideline representation on three specification levels: abstract, computable, and implementation. Abstract flowchart level allows conceptual modeling of guidelines and primarily accommodates human readability of GLIF guidelines. Computable specification is algorithmic and can be validated for consistency and logical completeness. The implementation level is the most specific and is tailored toward the integration into a concrete clinical system, thus may not be shareable. Additionally, GLIF supports encoding of structured medical vocabularies such as those that are part of the Unified Medical Language System (see Section 4.2.2).

An important interoperability feature of GLIF is that it includes a model for defining medical data based on HL7 Reference Information Model (see Section 4.2.1.1). Furthermore, unlike Arden, GLIF3 includes a specification of a formal query and expression language, Object-Oriented Guideline Expression Language (GELLO) [21].

GELLO supports the object-oriented model of GLIF and expresses its functions as class methods. GELLO provides support for such constructs as data types, collections, utility classes for logical operations, etc. It also is used for definitions of decision criteria and patient states in a guideline. GELLO, in fact, is a superset of Arden's Syntax logic grammar and there is even a possibility to map GLIF-encoded guidelines to MLMs [15]. Therefore, GLIF can be considered a broader, more expressive extension of Arden guideline representation model. It is important to note, however, that GLIF lacks formal semantics which is a major drawback for its implementation.

Figure 3 demonstrates a sample GLIF guideline, which is a translation of the hyperkalemia guideline presented in the Section 2.1.1. Note that this figure represents not the entire guideline, but only the graphical representation of the guideline algorithm.

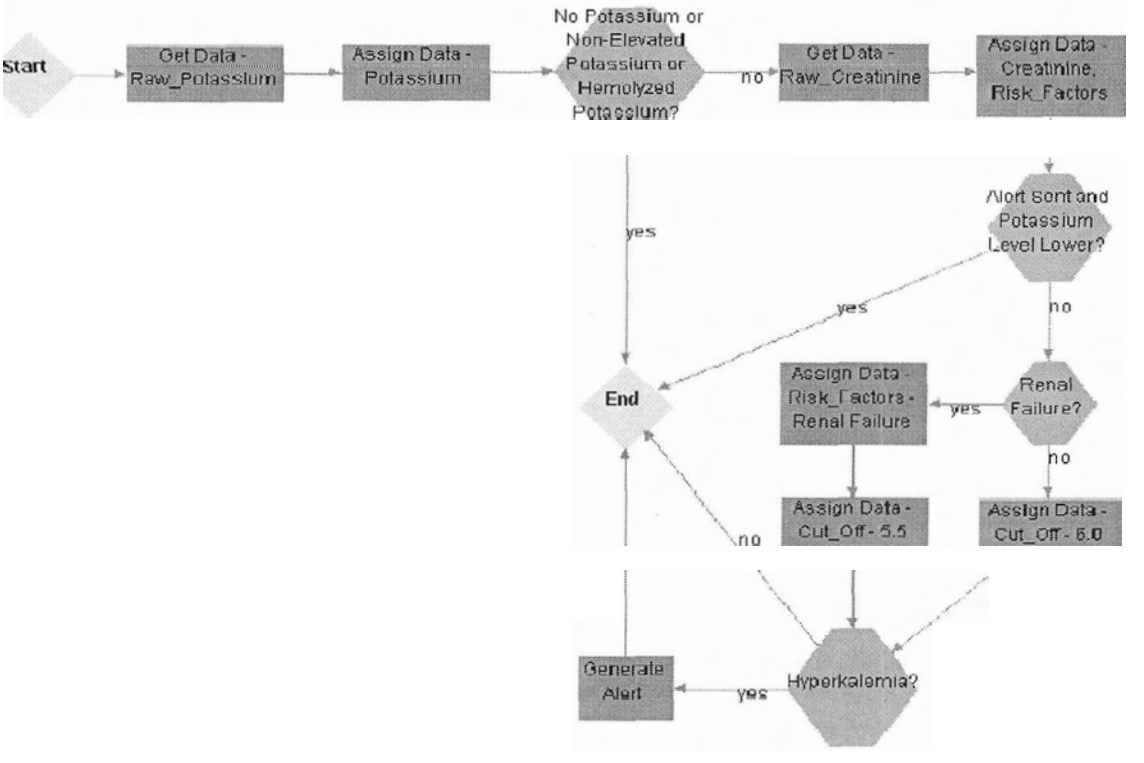

Figure 3: GLIF guideline

Current software availability for GLIF includes Protégé-2000 based authoring and validation tools. GLEE [22] has been frequently mentioned in the literature as the GLIF execution engine; unfortunately it is not currently available outside the InterMed group. It is interesting to note, however, that HL7 has taken a keen interest in both Arden and GLIF. This strategic decision is likely to ensure active development of these languages and perhaps even wider adoption in healthcare industry.

2.1.3 PROforma

PROforma [23] is another formal knowledge representation language. It was introduced in 1992 by the Advanced Computation Laboratory for Cancer Research in the United Kingdom. PROforma combines logic programming and object-oriented modeling and is based on Reinforcement Learning (RL) language. RL is a subset of C language with additions for state machines; RL is often employed in artificial intelligence domain. Moreover, PROforma's syntax is expressed in the Backus-Naur Form (BNF) and its operational semantics is expressed in a language similar to Z [24].

PROforma language models a guideline as a set of task and data items, where all tasks are derived from the generic task and are separated in four categories: plans, decisions, actions, and enquiries. Plans organize hierarchies of tasks; actions represent procedures that must be executed in the external environment (e.g. performing a surgery); enquiries indicate points in the guideline where additional information must be acquired for the guideline execution to proceed; decisions represent choices either about what to do or what to believe [25].

All tasks in PROforma share attributes that describe goals, control flow, preconditions, and post-conditions [26]. The intention of PROforma is to create a simple yet expressive

model with a minimal set of constructs. Such straightforward approach facilitates learning and use of the PROforma language.

Figure 4 demonstrates a sample PROforma guideline [25] for suspected breast cancer referral.

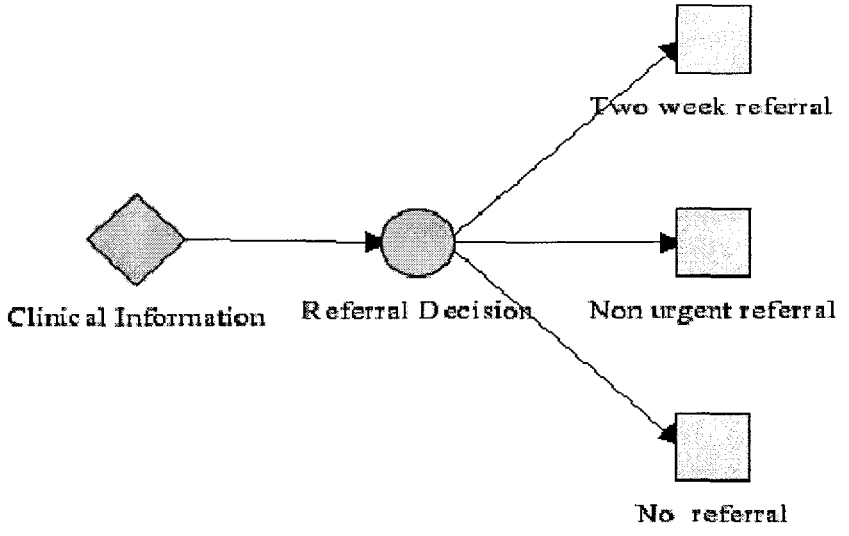

Figure 4: PROforma guideline

Currently there two main implementations of PROforma authoring and execution engine: a commercial tool Arezzo ® and a free tool called Tallis which is available only for academic use and evaluation. In addition, a large number of clinical applications was built on these technologies primarily in UK.

PROforma stands out from other guideline representation languages as an expressive language that was successfully implemented in clinical settings and has a promising perspective to be expanded, improved, and implemented on an even wider scale. A drawback of PROforma is that it does not yet comply with existing standards of data

structure and interchange such as HL7 RIM (see Section 2.2.1.1), or XML. In addition lack of open source tools also impedes PROforma adoption.

2.2 Clinical Information Standards

Standard compliance is important for interchanging clinical information, such as patient records, among distributed EMRs and CDSSs. Here we discuss several major initiatives that foster developments in standard clinical information structure and interoperability, namely RIM and CDA by Health Level 7 (HL7) organization.

2.2.1 Health Level 7

One of the major organizations that advance the development of healthcare information standards is Health Level 7 (HL7) [27]. It provides guidance in information structure and integration.

HL7 is an internationally recognized, ANSI-accredited standard developing organization, operating in the healthcare domain, producing specifications for clinical and administrative data representation and interchange. The term "Level 7" refers to the highest level (application level) of the Reference Model for Open Systems Interconnection (OSI) defined by International Organization for Standardization (ISO) [28]. In the recent years HL7 standards have moved to an object-oriented and document-centric paradigm of clinical information management. HL7 Reference Information Model (RIM) is in continuous development and has been incorporated into other standards such as Guideline Interchange Format (see Section 2.1.2). Furthermore, HL7 has released the Clinical Document Architecture (CDA) as a standards proposal for representating of

clinical documents. Both of these specifications address the need for clinical information standardization and are described here in more detail.

2.2.1.1 Reference Information Model

The HL7 RIM is an ontology of health related information concepts, which attempts to provide a common semantic basis for defining specialized data structures for specific data domains. RIM is expressed as a set of classes and relations among them. In addition, HL7 also defines the data types as well as structured vocabularies for coded RIM attributes. Domain-specific information models (DIMs) and Refined Message Information Models (RMIMs) can be derived using the RIM as a meta-model. An example of an RMIM is the CDA model.

2.2.1.2 Clinical Document Architecture

CDA is an HL7 standard for representation of clinical documents using XML-format [29]. CDA became an ANSI-approved HL7 standard in 2000 and is currently at its release 2.0. CDA aims to facilitate interoperability and information exchange across heterogeneous clinical systems. RIM serves as a basis for CDA and thus ensures the use of common health information terminology.

CDA specifies the structure and semantics of data included in a clinical document. A CDA document may carry coded data, free text, multimedia artifacts, or a combination of these. The documents are intended to be both human-readable and computer-interpretable.

Any CDA document must contain a header and a body. A CDA header carries identification and administrative information for the document. The body may be

unstructured or structured as sections, each of which has entries for representing fully or partially coded content.

The CDA specification defines three levels of CDA compliance. Level 1 requires a coded header and permits text content with simple formatting of sections. Level 2 indicates a coded header and standard codes for sections. Level 3 CDA document, on the other hand, is fully structured with all data elements derived from RIM. "Interoperability by design" is the major goal of CDA standard. It provides a framework for representing various types of clinical documents therefore it is a broad and reasonably comprehensive specification. On the implementation level, however, the richness of CDA must be constrained to define domain specific document models which are referred to as CDA templates.

CDA templates prescribe the required structure for specific document types such as referrals, recommendations, etc. Templates can also define the use of certain external medical vocabularies for coded values.

2.2.2 Medical Terminologies

In order to codify clinical knowledge such as guidelines or patient information, it is necessary to utilize standardized medical terminologies. In the medical community a large number of various coding systems have been developed to address the specific needs of different sub-domains such as disease management, laboratory data, pharmaceuticals, etc. Consequently, the diversity of available terminologies becomes a significant obstacle in data sharing among different clinical systems and institutions. While it is not possible to enforce one single biomedical classification vocabulary, the only feasible solution is to mediate and translate concepts from the plethora of

classification systems. For this purpose Unified Medical Language System (UMLS) was developed by the National Library of Medicine [30].

2.2.2.1 Unified Medical Language System

UMLS is not a standard in itself, but rather a cross-referenced collection of standards. It is an effort for unification of terminologies related to health and biomedicine. UMLS is intended to facilitate interoperability of clinical systems by providing a common reference to multiple classification systems. Moreover, UMLS establishes relations among concepts across different classification system.

UMLS knowledge sources are not targeted for any specific type of application and incorporate information from many fields of healthcare and biomedicine, including guideline ontologies. UMLS allows heterogeneous clinical systems to "understand" each other even if they internally use different classification system for medical terminology. This is made possible by the UMLS software tools and knowledge sources: the Metathesaurus, the Semantic Network, and the SPECIALIST lexicon.

Metathesaurus is a very large knowledge source which contains over 1 million biomedical concepts and 2.8 million concept names from more than 100 controlled vocabularies and classifications. Some examples of classifications used by Metathesaurus are "Systematized Nomenclature of Medicine" (SNOMED) [31] and "Logical Observations, Identifies, Names, and Codes" (LOINC) [32]. This multi-lingual vocabulary database also defines relations among concepts across multiple terminologies. All Metathesaurus concepts are linked to at least one semantic type in the UML Semantic Network.

Semantic Network provides consistent categorization of concepts by defining semantic types through textual descriptions and their hierarchical relationships. The SPECIALIST lexicon provides lexical information on UMLS biomedical vocabularies to the SPECIALIST natural language processing system.

One of the software tools developed under UMLS umbrella is UMLS Knowledge Source Server which provides internet-based access to the knowledge sources for remote users such as individuals and software programs.

UMLS software tools and knowledge bases are multi-platform, freely available, and can be further customized by developers and information specialists for specific system needs.

2.3 Clinical Decision Support Systems

This section describes several prominent clinical decision support systems that were developed over the last several decades. These systems were designed according to their particular methodologies, with emphases on specific functionalities, and have shown different rates of success. However, examining this related work provides a valuable insight into initiatives that sprung in the domain of clinical decision support.

2.3.1 Leeds

Leeds system was developed in the late 1960s as a computer-based decision aid for diagnosing the cause of acute abdominal pain [33]. The reasoning behind Leeds was based on Bayesian theory (see Section 2.2.4) and built-in assumptions. Interestingly, in one of the case studies [34] where Leeds decision making was tested, the system reached 91.8 percent accuracy in diagnosing abdominal-related problems for several hundreds of

emergency patients. In the same study, clinicians, on the other hand, were able to correctly diagnose only 65 to 80 percent of cases. However, although Leeds was adopted in many emergency departments, it never reached the same high degree of accuracy in other clinical settings that it did in the above mentioned study. As suggested in [35], possible explanation for this phenomena is that clinicians may have differently interpreted such data as sensitivity or symptoms before they entered them into the Leeds system.

2.3.2 MYCIN

MYCIN was probably the most famous early decision support system [36]. It was designed for diagnosing infections and recommending antibiotic therapies.
MYCIN is a goal-directed system that represents clinical knowledge as sets of production rules --If-Then rules-- that relate observations to associated inferences [37]. MYCIN also has the ability to assign certainty factors to the derived diagnoses. By using backward-chaining reasoning, MYCIN engine performs a search of applicable rules that could potentially satisfy the desired outcome.

MYCIN is expert system shell under the PublicDomain license. It is an advanced but not sufficiently mature system for clinical use. One of MYCIN's shortcomings is the system's unawareness of its limitations when it provides a recommendation even if it does not have sufficient knowledge for making the recommendation [38]. Furthermore, MYCIN has a limited ontology and derives its decisions as a function of the information elicited about the patient, without making a prognosis of the treatment effect [38]. Although MYCIN was never used clinically, it served as a foundation for further development and research of clinical decision support systems.

2.3.3 HELP

HELP, short for Health Evaluation through Logical Processes, is an integrated hospital information system with strong decision support capabilities [39]. It was designed for clinical, teaching, and research purposes at the University of Utah in 1970s. It allows specialization of decision logic and its close integration with patient data. HELP's event-driven approach facilitates system reaction to newly entered patient data for generation of alerts and suggestions. HELP also has the ability to periodically monitor the repository of patient data and perform specific evaluations. Easily understandable HELP Frame Language can be utilized directly by the clinicians for writing various protocols for analyzing data and collecting interesting statistics. HELP also incorporates Arden Syntax (see Section 4.3.1) as a standard formalism for specifying decision rules. Because of HELP's successful performance, it serves as a good example of how clinical systems can benefit from the decision support functionalities in real clinical settings.

Because HELP is not an open-source system, its decision support functionality cannot be leveraged in an open source CDSS proposed in this thesis.

2.3.4 EON

EON is a guideline modeling and execution framework developed by Stanford University and first introduced in 1996 [40]. It is not a stand-alone system and functions only as a part of a clinical information system that integrates EON framework. EON has "plug and play" architecture based on components that can be easily reconfigured for different functionalities. Three main EON component types are: problem solvers, knowledge bases, and database mediators. Problems solvers perform specific decision

making tasks; knowledge bases serve as repositories of clinical guidelines; database mediators provide interoperability between the native data sources (that store patient data) and EON components.

EON includes data models for representing domain ontologies, patient data, and guidelines. EON guideline algorithms are modeled in Protégé and are represented as sets of scenarios, action steps, decisions, branches, and synchronization steps [41]. In order to facilitate expressiveness of the encoded clinical knowledge, EON adopted such constructs as object-oriented language, temporal query and abstraction language, and first-order predicate logic.

The main advantage of EON framework is that it supports the reuse of medical knowledge, temporal queries, and abstractions. Another interesting feature of EON functionality is that it can provide not only recommendations, but also explanations and arguments of how a particular recommendation was derived.

EON system was used only experimentally and its success of adoption in a real clinical setting was not seen. In fact, currently EON project is not under development, but is partially carried out by a project called SAGE [42] which is a proprietary initiative.

2.4 Reasoning Methodologies for Decision Support

There is a variety of reasoning methodologies applied in the design and implementation of decision support systems. The goal of this chapter is to give a general overview of most prominent knowledge processing and reasoning techniques, which are distinguished here on two levels: symbolic and sub-symbolic [43]. However, it is important to know that described methodologies are not necessarily disjoint [44] [45].

2.4.1 Symbolic Reasoning

Symbolic reasoning [46] represents different types of knowledge such as facts, concepts, and rules through explicit symbols. Symbolic reasoning methodology defines but is not limited to rule-based and case-based models.

2.4.1.1 Rule-based Model

The systems that implement rule-based reasoning are often referred to as expert systems. The foundation for modern expert systems was outlined by Newell and Simon [47] in the early seventies when they proposed a production system model. In this model the knowledge, called production rules, is persistently stored by the system, while the problem-specific facts are stored in a short-term memory. The rules are formulated as sets of predefined If-Then statements which can represent relations, recommendations, strategies, directives, and heuristics. These rules must be defined by the domain experts in advance. During the execution, the inference engine then applies production rules to the given facts and derives conclusions. An expert system may also have the ability to present explanations how a certain conclusion is reached and why specific facts were needed.

Rule-based systems can be differentiated as those that offer forward or backward chaining. The chaining itself refers to the technique for matching the rules to the provided problem facts. In forward chaining, the reasoning starts from the known data and proceeds forward by evaluating each rule. When the "if" condition of a rule is satisfied, the rule is fired and a new fact is derived and stored. This process continues

until the rule base is exhausted. This approach of forward chaining is also called data-driven reasoning.

For situations where there are both a problem and an established goal to infer a particular fact, the forward chaining approach is not efficient. Therefore a backward chaining, or goal-driven, technique can be applied. Typically, in backward chaining the rule base is first searched to find the rules that satisfy the predefined goal (hypothetical solution) in their "then" part. When an applicable rule is found, its conditional ("if") part is evaluated and the rule is fired or discarded accordingly. An example rule taken from a hyperkalemia guideline may look as shown in Figure 5:

Rule: Hyperkalemia Chek
IF <potassium level => 4.7>
THEN <conclude hyperkalemia>

Figure 5: Clinical rule example

The above rule evaluates a patient's whole blood potassium value, and if the level of potassium ions is elevated then hyperkalemia is concluded.

Because a backward chaining mechanism works with a desired goal "in mind", it is more complex than forward chaining, but also more efficient for automated decision support as it considers only rules that lead to the specified goal. On the other hand, forward chaining may have efficiency advantages as it derives facts and uses them only once unlike in backward chaining where derived facts are not stored and are re-evaluated each time they are produced. However, the efficiency of a particular reasoning system depends largely not on the reasoning technique itself but on its implementation (algorithm). One of the most efficient algorithms used for rule-based expert systems is Rete [48]. Rete (from Latin for "network") builds a network of nodes where nodes

represent patterns in the left-hand-side of the rule [49]. These nodes constitute a decision tree that combines all the patterns in the knowledge base. The given facts flow through the Rete graph networks and are matched to the rule patterns. Rete sacrifices memory for increased speed and is theoretically independent of the number of rules in the knowledge base. However, in very large expert systems Rete is known to run into memory consumption problems. Other Rete-based algorithms have been designed to address this issue.

Although it may seem that rule-based reasoning is straightforward, it is not necessarily always so. A common problem with rule bases is the presence of conflicting rules in the same system. For example, there may be two rules that have the same "if" part, but different "then" parts. In such cases, various conflict resolution techniques should be applied [50].

2.4.1.2 Case-based Model

Case-based reasoning [51] is another symbolic reasoning approach [52]. Case-based models have been applied in medical domain because they closely resemble the process of experience-based decision making by a clinician.

Case-based reasoning uses previous solutions to solve similar or new problems. The knowledge of the case-based system is embodied in the library of past cases, rather than encoded as rules. Each case consists of a problem description, solution, and outcome. The reasoning process itself is not explicitly expressed, but is implicit in the solution. In order to solve a given problem, it is first matched to the most similar cases in the case library. These cases then are analyzed and are used to suggest a solution. The current problem reuses the solution, which can be tested and revised. Finally the given problem

and its applied solution get retained as a new case in the case library. There are many methods for organizing, retrieving, and analyzing the knowledge represented in past cases. The heuristic nature of such knowledge of course dictates the use of semantic interpretation although some less functional systems retrieve cases merely on syntactic similarities.

The two commonly used algorithms for case-based knowledge systems are nearest-neighbor retrieval and inductive retrieval. Nearest-neighbor technique [53] is simple as it computes the similarity between the new case and the stored cases based on weight features. Inductive retrieval algorithm [54] determines which features are best for discriminating cases and generates an in memory decision tree. Case-based systems may also implement these algorithms as complimentary where the best matching cases are selected with inductive retrieval and are ranked by their similarity to the new case by nearest-neighbor algorithm.

The challenge associated with case-based systems is adapting the cases and solutions in order to solve certain problems that are not closely analogous to the recorded cases [55]. Thus often a case-based system serves more as a reference than an advanced reasoner.

Rules and cases are relatively simplistic ways of expressing real-world knowledge. Even though expert systems can be built on the rule-based and case-based models, they typically have to be narrowly specialized for particular domains as they are not able to generalize or manipulate non-explicit knowledge. For this reason, a number of other techniques for problem solving have been developed as discussed in the next section.

2.4.2 Sub-Symbolic Reasoning

Decision making systems based on symbolic reasoning process knowledge by sequentially applying predefined logical rules or analyzing predefined cases. Therefore it is a prerequisite that such rules/cases be defined or collected in advance. Such approach may work well for some applications, however the complexity, fuzziness, and unpredictability of clinical knowledge requires a more advanced approach such as sub-symbolic reasoning [56] where explicit pre-formulation of all applicable rules and conditions is not required. Sub-symbolic knowledge representation is often applied to information problems when meaning is uncertain or probabilistic. Sub-symbolic reasoning falls in the domain of artificial intelligence (AI) which is a subfield of computer science that manipulates abstract concepts. Sub-symbolic reasoning models have a connectionist structure that has been trained, not completely predefined. These models store knowledge in a distributed form and are best represented as networks. The reasoning then is usually represented as the adjustment of weights on the network's nodes.

Here we discuss a common type of sub-symbolic reasoning model: artificial neural networks (ANNs) as adaptive and self-learning systems.

2.4.2.1 Artificial Neural Networks

With ANNs, the rules of data processing do not have to be specified before the data is processed. Rather, the rules are dynamically derived by the adaptive and self-learning ANNs.

ANNs are also known in software engineering field as connectionist systems, parallel distributed systems, and adaptive systems. ANNs are defined as "...computational paradigms based on mathematical models that unlike traditional computing have a structure and operation that resembles that of the mammal brain." [57]. ANNs are designed as models of processing elements that operate in a parallel and decentralized fashion.

While at the present state ANNs are far from simulating complex computational properties of the brain, they have been sufficiently developed as advanced mathematical algorithms for automated reasoning in many application domains. In health care ANNs are gaining momentum for performing such tasks as pattern recognition, optimization, compression of medical data, and of course clinical decision making [58].

Neurons are basic processing elements of ANNs. Each neuron is capable of receiving input, processing it, and sending output to another connected neuron. The output produced by a neuron must be limited to a well-defined range and is determined by the neuron's weight of the input and its internal function. Typically ANNs are arranged in layers where each layer is represented by an array of neurons. For example each ANN should have at least an input and output layer with any number of layers in between. In conjunction, layers of neurons work together and can perform very complex tasks.

There are two models of behavior for ANNs: learning and testing. During the learning phase ANNs process series of problems and "guess" the solutions. With time and possibly feedback, however, ANNs begin to adapt internally and learn how to solve problems to reach satisfactory outcomes. A common algorithm applied for training an ANN is back-propagation algorithm [59], which calculates an error derivative of the

weights assigned to the nodes. The adjustment of weights allows minimization of error between the desired and actual ANN output.

When the system reached certain performance level, it operates in a testing phase where the learned problems are generalized and new ones can be processed based on ANNs learned "skills". With such approach ANNs can be trained to address complex problems and operate on vast amounts of knowledge.

Some drawbacks of adopting ANNs for designing a decision support is finding proven ANNs for specific problem domains [60].

2.4.2.2 Bayesian Networks

Medical decision making is often associated with uncertainty in respect to observations, findings, recommendations, diagnosis, and disease management. Moreover, conditional probability, where one event is conditional on the probability of a previous one, is often evident in a clinical setting. Therefore application of probabilistic theory such as the one used by the Bayesian networks is well suited.

An advanced form of computability, a Bayesian network [61] is a directed graph that is based on probability theory, primarily on Bayes' theorem. BN graph includes nodes, arcs and tables. Nodes represent uncertain variables and are associated with the tables that store probability distributions. The conditional probabilities are typically estimated by domain experts at the time of network design. This approach of allowing the introduction of prior knowledge differentiates Bayesian inferences from other reasoning models. For the discussion on Bayesian network robustness algorithms see [62].

A limitation of Bayesian networks relates to the quality of the prior beliefs used in the inference processing. A Bayesian network is only as useful as this prior knowledge is reliable [63].

With all their benefits and implications, Bayesian networks are becoming an increasing large area of research and application in AI field. Moreover, Bayesian networks have already shown to be successful in clinical decision support [37]. As mentioned in Section 4.1.1.1, a large experiment on applying Bayesian networks for decision support was done by deDombal in Leeds system. Interestingly, Bayesian methods have also been used for artificial neural networks [64].

2.5 Interoperability of Distributed Service-Oriented Systems

As the need for electronic collaboration in the health care domain is growing [65], increasingly complex problems in interoperability are arising [66]. Health organizations have numerous legacy systems that are not designed to interoperate but must be included in information sharing processes of today [67]. Hence, there is an array of heterogeneous distributed systems that offer software services and require integration. The term "service" refers to a self-contained and well-defined function that is provided by an underlying computer system.

In order to facilitate the interoperability among these systems and to provide integrated service-oriented architecture (SOA) [68], a middleware layer must be introduced. Middleware is software that facilitates programmatic access to distributed software services, abstracts from specific system implementations, and provides standard communication mechanisms [69].

Because this thesis proposes a CDSS that is independent of clinical systems and provides a value added service, one of the foci of our work is the interoperable middleware for distributed SOA systems. Therefore, next we describe and evaluate several technologies for implementing SOA.

2.5.1 Technologies for Service-Oriented Architectures

There are several major technology platforms for engineering distributed systems in SOA paradigm including DCOM [70], EJB [71], RMI [72], CORBA [73], and Web Services. While DCOM, EJB, and RMI are designed to be interoperable, they are geared towards specific technology groups such as Microsoft for DCOM and Java for EJB and RMI. CORBA and Web Services, on the other hand, are designed to be both platform and language independent. Therefore, in order to achieve the maximum interoperability for our CDSS, we have identified the latter two middleware technologies as the candidates for implementing the SOA for our system. The following sections describe both CORBA and Web Services in detail.

2.5.1.1 CORBA

Common Object Request Broker Architecture (CORBA) is an open standard for distributed object computing defined by the Object Management Group (OMG) [74]. It is intended to be a middleware technology that is agnostic of the underlying platform, operating system, language, network interconnections, and hardware [75].
CORBA applications are composed of client and server objects that are mediated by the Object Request Broker (ORB). ORB-to-ORB communications, in turn, take place over OMG's standard Internet Inter-ORB Protocol (IIOP).

CORBA objects expose their functionality by means of the Interface Definition Language (IDL) which is similar to C++ syntax. Both the client and server have to share the same IDL that is compiled into a stub and a skeleton respectively. The stub becomes a stand-in for the method being called remotely and the skeleton translates the remote invocations into method calls on the local object. Essentially, the interface represents a contract that the server object offers to the client.

Within the CORBA framework, interface meta information is maintained by the Interface Repository component. Platform, operating system, language, and location transparencies in CORBA are all enabled by separation of object's interface and implementation [76]. The IDL itself is language-independent but can be bound to different programming languages via OMG IDL specification. Today, there are a number of programming languages that support IDL mappings including C, C++, COBOL, Lisp, Java, Perl, Python, etc [77].

Encapsulation of the object data and code with the communication mechanisms exposed on the interface level allows a client and server to be compiled in different languages and running on different platforms. Furthermore, CORBA object references do not disclose to the client whether the server object is local or remote, which ensures location transparency.

Figure 6 illustrates a CORBA model that enables the interoperability and transparencies described above.

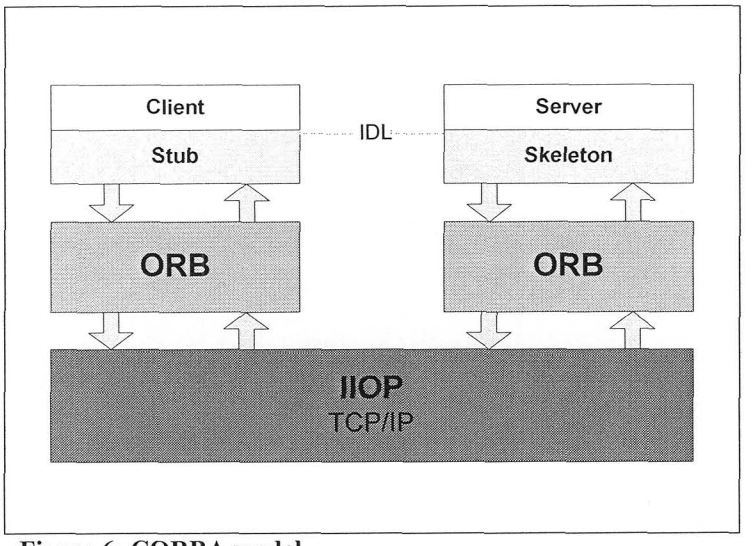

Figure 6: CORBA model

2.5.1.2 CORBA Services

CORBA has a number of complementary OMG specifications for higher-level services that provide extended functionalities in a standard environment [78]. Each service specification is defined as an IDL and builds on these major fundamental CORBA principles [79]:

- Separation of interfaces from the implementations
- Typing of object references by interfaces
- Use of multiple inheritances of interfaces
- Use of subtyping for extensions and specialization of functionalities

Individual services are relatively simple, but they may be combined to support advanced functionalities. For example, event, life cycle, and future relationship service can facilitate CORBA object graphs [80]. Individual service specifications, however, are designed to be generic and suitable for both local and remote implementations. In addition, to provide the flexibility of implementation, services are decomposed into distinct interfaces for different service clients. Therefore, a service can be implemented

as one or many objects to support different specified interfaces. The major CORBA services that directly enhance the interoperability among distributed systems are: Naming Service, Trading Object Service, Transaction Service, and Security Service. Further information on CORBA services can be found in the Catalog of OMG CORBAservices Specifications [81].

2.5.1.2.1 Naming Service

The Naming Service facilitates association of symbolic names with specific objects [82]. It allows objects to be registered and located by name. This service is a principal mechanism though which clients locate objects that they wish to use. This service is commonly provided in many ORB implementations because finding a service is orthogonal to using it.

2.5.1.2.2 Trading Object Service

The Trading Service works as a "yellow pages" service by providing a registry of objects with their properties [83]. Similarly to the Naming Service, it allows one to locate CORBA objects. However, instead of searching by a specific name, a CORBA object can be retrieved based on its characteristics such as operation names, parameters, or result types. In fact, most CORBA implementations include both Naming and Trading services [84].

2.5.1.2.3 Transaction Service

Transactions are an important part of building reliable and available applications. CORBA Transaction Service specifically addresses the concept of distributed transaction

processing. It provides interfaces for supporting transaction capabilities and allows an ORB to perform the function of a distributed transaction processing monitor [85].

2.5.1.2.4 Security Service

CORBA Security Service specifies the interface for security features such as identification and authentication of users, auditing of user's actions, authorization and access to services, non-repudiation, security of communication, and administration of security policies. It is a reference model that provides the overall framework for CORBA security [86] and remains security technology neutral.

2.5.1.3 Web Services

According to the Web Service Glossary offered by the World Wide Web Consortium (W3C), a Web service can be defined as "a software system designed to support interoperable machine-to-machine interaction over a network" [87]. More precisely, Web Services are self-contained programmatic interfaces that encapsulate distributed components and allow them to interoperate over Internet regardless of their internal heterogeneous structure. Standard Web Service communication protocols allow for loose coupling of such components [88], where interoperability among them is not specified during the design-time, but is dynamic and can be facilitated through run-time service discovery and compositions.

The Web Service architectural foundation is similar to CORBA [89], where the middleware layer facilitates the transparencies of objects' location and underlying implementation. Similarly to CORBA, the client acts as if it is invoking methods on the local object instance, while in reality it is directly communicating with the service proxy

on the client side. The invocation propagates through the network and is executed on the remote server object itself.

2.5.1.4 Web Service Stack

Web Services encompass a stack of complimentary specifications that are developed to address different levels of interoperability, as illustrated in Figure 7. In the following subsections we describe each layer of the Web service stack [90] in greater detail.

Layer	Standard
Semantics	DAML-S, OWL-S
Service Flow	BPEL4W
Service Discovery	UDDI
Service Description	WSDL
Secure Messaging	WS-Encryption, WS-Signature, SAML
Messaging	XML, XML Schema, SOAP
Transport	HTTP

Figure 7: Web Services stack

2.5.1.4.1 Transport

The most common transport used with Web Services is Hypertext Transfer Protocol (HTTP) [91] which employs TCP/IP as the underlying network communication protocol. HTTP resides at the base for the Web Service stack and is widely deployed. It is a payload-agnostic transport and offers similar connection management features of CORBA. It is important to mention that Web Services can also leverage other transport

protocols [92] such as Secure HTTP (HTTPS), Reliable HTTP (HTTPR), File Transfer Protocol (FTP), or Standard Mail Transfer Protocol (SMTP).

2.5.1.4.2 Messaging

On the messaging layer there are the Extensible Markup Language (XML), XML Schema, and Simple Object Access Protocol (SOAP).

XML [93] is a platform-neutral common data representation language. This simple and flexible text format facilitates data serialization into messages for their exchange over the Web. XML Schema [94] is a related standard for expressing vocabularies and predefined rules such as structure, content, and semantics for expressing XML data and processing of XML documents.

SOAP [95] is a standard for structured exchange of XML documents. It is a W3C recommendation proposed by a group of companies such as IBM, Microsoft, UserLand, DevelopMentor, and Lotus. More specifically, SOAP is a lightweight XML-based protocol and a messaging layer for exchange of information over the Internet. SOAP envelopes encode HTTP request and response parameters of method invocations. Moreover, SOAP presents an extensible framework for defining the structure and constructs as well as rules for creating and processing messages, which can be exchanged over a variety of underlying communication protocols [96].

2.5.1.4.3 Secure Messaging

Some of the major decentralized security models leveraged by Web Services are W3C approved XML Signature [97] and XML Encryption [98]. XML Signature embodies syntax for representing signatures in Web resources and procedures for computing and

verifying such signatures and is intended for encrypting/decrypting digital content of a message exchange between parties. XML Encryption, on the other hand, specifies the content itself as well as the information needed to needed to decrypt it.

An alternative Web Service standard for the exchange of authorization and authentication information is Security Assertion Markup Language (SAML) [99]. This language differs from other standards by utilizing a model of assertions for establishing trust among collaborating entities [100].

2.5.1.4.4 Service Description

Web Service Description Language (WSDL) [101] is an XML format for describing Web services and messages exchanged between them. This specification is a merger of two earlier standards: Web Interface Definition Language (WIDL) [102] and Network Accessible Service Specification language (NASSL) [103].

WSDL is used to separate abstract service descriptions from the specific deployment of a service [104]. The WSDL file contains information necessary for a client to interact with the service, including location of the service, publicly available operations, data types included in the messages, and the required communication protocol. In addition, WSDL specifies the format and content of the request and response messages of a given Web Service [105].

2.5.1.4.5 Service Discovery

Web Service publication and discovery for e-business integration is enabled by the Universal Description, Discovery and Integration (UDDI) specification [106]. This cross-industry effort, by a group of thirty six industry leaders including IBM and

Microsoft, provides a platform for Web Service directories.

The UDDI specification defines a model of entries associated with service and business descriptions, as well as a search API to query for such entries [107].

2.5.1.4.6 Service Flow

Service flow refers to the utilization of Web Services for distributed workflows. There are several proposed specifications for defining how collections of Web Services can be jointly used to realize complex functionalities of business processes. From the overview and comparison of these specifications done by Wil van der Aalst [108], Business Process Execution Language for Web Services (BPEL4WS) stands out as the more notable standard for service composition.

BPEL4WS [109], or BPEL for short, is a process modeling language used for composing services according to specific business rules. BPEL is intended for modeling two types of processes: abstract and executable. An abstract non executable process specifies message exchange behavior without revealing the internal behavior of any services. An executable process defines partners, messages, and execution order of activities within a process. BPEL uses WSDL for describing the process interfaces of the partner services. The message exchanges within a BPEL process take place through these interfaces in a peer-to-peer manner. Moreover, such abstract interface definitions of the partner services ensure platform and transport independence.

In addition, BPEL also provides fault handling and compensation functionality and can also be extended with WS-Transaction and WS-Coordination to provide robust service transaction mechanisms [110].

2.5.1.4.7 Semantics

In order for Web Services to be automatically interpreted by intelligent agents, Web Service must be described with formal machine-interpretable ontologies [111]. This requirement to encode meta data on the Web led to the emergence of several formal ontologies for describing capabilities and properties of Web resources and Web Services in particular. These formalisms include DARPA Agent Markup Language for Web Services (DAML-S) [112] and Ontology Web Language for Web Service (OWL-S) [113], which both build on the earlier work of the Resource Definition Framework (RDF). More information on the evaluation of ontologies can be found in [114].

Chapter 3 Requirements

3.1 Knowledge Interoperability Requirements

In this section we identify requirements for a guideline language as a facility for clinical knowledge representation, specifically in the domain of preventive care. A review of guideline language characteristics has been done by [115], however the focus of the authors there was mainly on the structural guideline attributes. Our requirements, on the other hand, extend the scope to include other important aspects of guideline languages.

3.1.1 Open Content License

This thesis emphasizes an open source CDSS framework; therefore the selected guideline representation language must have an open license. Openness of the specification, can be ensured by an explicitly stated license such as GNU Free Documentation License [116] or an equivalent. In addition to the legal compatibility with an open source CDSS, an open guideline language has benefits to be objectively evaluated for its expressiveness, reasoning, and problem solving methods.

3.1.2 Formal Semantics

Formal semantics should be defined in order to provide unambiguous interpretation of the guideline language constructs. Formal semantics are important for verification of the language properties such as termination of a guideline instance, resolution of conflicts in

a guideline, etc. It is also important that the semantics be clear and not overly complex to allow a guideline encoder to reason about the guideline behavior at the design time.

There are a number of ways how semantics could be formally specified [25]. The following list gives examples of acceptable formal methods [117], but is not constraining for the choice of a guideline formalism.

- Denotational Semantics

  The meaning of the languages is defined in terms of mathematical functions. Using the standard mathematical theory of functions, the language is mapped to mathematical functions whose properties can be proven [118].

- Axiomatic Definition

  A set of assertions about properties of the language constructs constitute axiomatic semantics.

- Translation

  The semantics are defined in terms of rules that define how the language can be translated into another language whose semantics are known [119].

- Extensible Definition

  The language semantics are defined as primitive operations.

- Operational Semantics

  A set of rules specify how the states of an abstract machine change with the enactment of operations [120].

3.1.3 Formal Syntax

A guideline language should have a syntax definition formalism. The most common syntax definitions are formed by context-free grammars [121] which are particularly

widely used in computer science. Commonly used notations for context-free grammars are Backus-Naur Form (BNF) and its variations [122].

As part of the language syntax, expression constructs should be supported in order to perform queries on the patient data, reference medical concepts, and express temporal relations

The benefits of having structured semantics and syntax go beyond implementation aspects of a language. The availability of formal definitions can allow independent evaluation of guideline languages. The importance of formalizing a guideline language is discussed in detail by [115].

3.1.4 Expressive Decision Model

A guideline language should support an expressive decision model as it is the key aspect of decision support. Expressiveness can refer to the ability of a language to represent advanced guideline constructs such as branching, parallel execution, synchronization, etc. These properties closely relates to the formal semantics of the guideline language. In addition, an expressive decision model also addresses the decision model from the reasoning methodology point of view, as elaborated in this particular requirement.

The expressiveness of the guideline language should reflect the intended application of the language. In other words, different degrees of expressiveness should be required for representing highly complex and interactive guidelines as opposed to relatively simple preventive care guidelines which are addressed in this thesis.

In the area of clinical practice guidelines, several decision methodologies have been applied [115] such as rule-based, case-based, logic-based, decision analysis, etc.

For the purpose of this thesis which addresses preventive care and reminder-based decision support, we have identified rule-based and logic-based methodologies to be most appropriate.

- Rule-based methodology

Rule-based decision methodology is one of the earliest to be applied in clinical decision making [123]. The benefit of this approach is that it is relatively straightforward because rules represent common logic and can be easily understood by the software engineers and clinical domain experts. Moreover, the overwhelming majority of guidelines that are in clinical use today are represented in a rule-based fashion. Therefore, adopting a guideline language with a rule-based methodology is a feasible transition from existing free-text to computer-interpretable guidelines. We selected the rule-based methodology for encoding preventive care guidelines of low and moderate complexity.

- Logic-based methodology

We identified logic based methodology as a requirement for encoding large and highly complex guidelines. With this approach, guidelines are stored in a declarative knowledge base and their selection is determined by logic criteria.

Logic-based languages are more complex to apply to guideline modeling and execution. However, the advantage of using the logic-based formalisms is that they have an underlying mathematical model constructed on the predicate calculus. This allows one to prove that a given knowledge base is non-contradictory, non-redundant and minimally complete. In addition, logic-based algorithms can also be semantically validated.

3.1.5 Integration with a Standard Patient Information Model

The structure of the patient information data items referenced by the guidelines should be defined in accordance to a standard information model such as RIM (see Section 2.2.1.1) or equivalent.

Fixing the structure of the patient data leveraged for the guideline modeling and execution facilitates mapping of the CDSS input and output to the Electronic Medical Record (EMR) systems. In fact, it has been recognized that one of the major obstacles in establishing a widespread acceptance for guideline languages is the lack of their integration with standard patient-centric information models [124].

3.1.6 Integration with Standard Medical Ontologies

A guideline language must provide mechanisms for referencing standard medical ontologies such as drug and disease classifications. The language, at the same time, should be agnostic of any particular medical ontology.

The use of standard medical ontologies ensures that encoded medical concepts are not institution-specific and interoperable (see Section 2.2.2).

3.1.7 Conformance to Other Existing Standards

The guideline language should leverage the existing information representation standards in order to be adaptable to the current technologies. In addition, conformance to such information standards as HL7 RIM (see Section 2.2.1.1) can greatly facilitate interoperability and exchange of knowledge. For example, RIM could be used for integration of a patient information model with a guideline formalism.

3.1.8 Broadly Known Formalism

In order to have a good potential for a wide-scale adoption and interoperability, the guideline language should be broadly known, well publicized, actively developed and maintained by the experts. Active involvement of various interest groups in the language also results in the implementation of guideline knowledge bases which can be reused to the benefits of a wider community.

3.1.9 Structured Recommendations

Recommendation statements are fundamental assertions of guidelines [125]. In order to be interoperable and computer-interpretable, they should be structured according to an appropriate standard. Although recommendations can be very diverse, [126] argues that different guideline languages organize their recommendations in a similar fashion.

To address interoperability of recommendation sets, they can be structured as collections of decisions or recommended activities, as suggested by [125].

3.2 Information Interoperability Requirements

Automated decision support is an information intensive process. Ensuring information interoperability in a distributed system setting is especially important for the framework described in this thesis where clinical information is interchanged between the CDSS and various EMRs. Thus here we define main requirements for an interoperable clinical information model that is one of the major enablers for information sharing.

3.2.1 Open Content License

All the information standards leveraged by the open source CDSS proposed in this thesis must be conformant to the Open Content License [127] or belong to public domain.

3.2.2 Standard Status

Leveraging ongoing work that is carried out by an active standards body is the key to selecting an information model that is interoperable and has a potential to be widely adopted. For our work, the standardization aspect is especially important because within our framework the patient information is interchanged among various heterogeneous systems such as the CDSS and EMRs.

When selecting an information standard that should have the potential to be widely adopted, it is also important to identify a standard body which is active and involves representation from multiple areas of clinical domain as well as different interest groups such as vendors, academia, health care providers, etc.

3.2.3 Defined Structure

The structure of the information model should be clearly specified. Its semantics should be expressed through such constructs as association and generalization relations among model elements. The attributes should have assigned data types and, where applicable, should draw values from the sets of predefined codes, or controlled terminologies.

Selecting a thoroughly defined model with unambiguous structure and semantics greatly facilitates a common understanding and interoperability of the interchanged information.

3.2.4 Content Specification Process

A single static information model cannot provide the needed level of detail for all areas of clinical informatics without being prohibitively large and complex. Therefore, a common information model should be specified as a meta-model and serve as a point of reference and a root for extensions.

Furthermore, a standard formalized process for meta-model refinement and content specification must exist in order to derive domain specific information models.
The process must require that all derived information structures are traceable back to the meta-model. Moreover, the process must ensure that the semantic and business rules of the derived structures do not conflict with those specified in the meta-model.
It is also important that the derived domain information models can be objectively validated against the meta-model for conformance.

3.2.5 Compatibility with Existing Tools

Clinical information is highly complex, and so are the structures that represent it. Therefore, using available tools that assist in rendering, modeling, and verifying the correctness of the information structure and semantics is especially efficient for software engineers and informaticians. For this reason we define it as a requirement that an information model accounts for compatibility with existing modeling and validation tools.

3.3 Process Interoperability Requirements

Process interoperability addresses the integration of CDSS functionality into the local EMR workflows for easy and uniform access to decision support services and consistent utilization of clinical practice guidelines.

Our objective of designing a stand-alone interoperable CDSS is to achieve effective process integration with EMRs while marinating loose service coupling. Reducing dependencies among systems allows them to evolve independently without disrupting or entailing extensive changes on each other. To facilitate such approach, the following requirements were identified.

3.3.1 Platform-Agnostic Middleware

A middleware technology (see Section 2.5) is a technical integration facility required for communication between distributed heterogeneous systems such as a stand-alone CDSS and various EMRs. The middleware technology must be neutral towards the underlying platforms because EMRs are deployed on a range of various hardware and operating systems. In the widely distributed setting where multiple EMRs are envisioned to collaborate with our CDSS framework, it is not feasible to resolve systems heterogeneity on the application level. Therefore, ensuring platform transparency should be delegated to the middleware layer [128].

3.3.2 Transaction Management

Multiple points of control in distributed systems must be coordinated by middleware's transaction facilities. In particular, transactional capabilities are required to enable

specification of the CDSS services and the management of communications between the CDSS and other systems, as defined by the Communication Protocol (see Section 3.3.4).

3.3.3 Document-Oriented Messaging

To minimize the communication overhead and complexity of the communication protocol, we have identified document-oriented messaging as a requirement for process interoperability. Document-oriented operation is one in which messages carry coarse-grained, self-describing, and self-contained document-like structures. Documents present appropriate granularity for communicating patient-centric clinical data which is logically aggregated in clinical systems as patient charts. In addition, exchanging minimally complete structured documents reduces the number of method calls needed to obtain data from an EMR and provide recommendations back.

Note that this requirement does not impose how clinical documents are technically transferred between a CDSS and an EMR: as serialized objects or XML documents. Additionally, it is desirable to utilize a middleware technology that is successfully adopted in the industry in order to integrate CDSS functionality with a range of EMRs that are currently in production.

3.3.4 Communication Protocol

In order to achieve predictable results of the information exchange between a CDSS and an EMR, it is necessary to define a communication protocol and its outcomes. The protocol should specify the precise structure and allowed sequence of the interchangeable information units. The protocol should conform to widely accepted interoperability standards, be EMR-agnostic, and accessible via CDSS interface.

Furthermore, when defining a communication protocol for a CDSS, it is important to take into account how EMR vendors perceive process integration. Thus the resulting protocol should be not only effective but also feasible for implementation and adoption by various EMRs.

Chapter 4 Approach

4.1 Knowledge Interoperability Approach

In order to select an appropriate guideline language, we have systematized the knowledge interoperability requirements presented in Section 3.1 and devised a comparison matrix for the objective evaluation of three major languages: GLIF, Arden, and PROforma.

4.1.1 Comparison Matrix

Table 1: Guideline Languages Comparison

	Open Content License	Formal Semantics	Formal Syntax	Expressive Decision Model	Integrated with Patient Information Model	Integrated with Medical Ontologies	Structured Recommen-dations
GLIF	not official	no	no	yes	yes	yes	no
Arden	yes	no	yes	yes	no	no	limited
PROforma	not official	yes	yes	yes	no	no	no

From the analysis shown in Table 1 and other related research of guideline formalisms, we have derived two important findings. First, to our best knowledge, there is no guideline formalism that satisfies all our knowledge interoperability requirements. Second, one guideline encoding language was identified to match our requirements the closest: Arden Syntax. Next sections make the case for this language and explain in more detail our approach to making it a sustainable solution for facilitating knowledge interoperability between our CDSS framework and heterogeneous EMRs.

4.1.2 Arden Syntax

The main advantage of Arden syntax is that it is simple and straightforward language for encoding clinical knowledge. It is especially appropriate for encoding preventive care guidelines, which are addressed in this thesis, because they often tend to be simple and straightforward as well.

While Arden syntax lacks a formal semantics specification, it has a well-defined syntax that can be easily understood and utilized directly by most clinical domain experts. Arden is also the only officially standardized guideline language. It has undergone a decade-long evolution and is being continuously refined under the HL7 umbrella.

Meanwhile, Arden is primarily suitable for modeling such guidelines where each represents recommendations concerning a single clinical decision encoded as an independent MLM (see Section 2.1.1). While it is possible to call one MLM from another, such approach would cross the boundaries of the Arden specification as there are no defined directives how to chain MLMs. For this reason, Arden has mainly succeeded in clinical systems that offer reminder-based decision support functionalities based on guidelines of low complexity [129].

With these considerations, we have identified Arden to be sufficiently formalized and expressive for implementing rule-based guidelines that do not require complex workflow structures. This approach is consistent with our CDSS framework, where we intended to start with preventive care decision support and build on it incrementally in the future work. Therefore, in our work we do not address the issue of facilitating advanced workflow-like guidelines with Arden.

4.1.2.1 Formal Semantics

In our comparison matrix, the only language with a semantics formalism is PROforma. However, it does not qualify for our CDSS framework as it is not an officially open specification.

Arden, on the other hand, does not specify rules for proving the correctness of its implementation. The best solution to this problem would be to develop a formal semantics specification for Arden. However, the complexity of this task is beyond the scope and focus of our work.

Therefore, we propose a practical approach to address the lack of formal semantics in Arden by providing an operational definition of an Arden engine through our open source implementation. In addition, we provide heuristics and guidelines for developing an implementation of Arden Syntax. The details of this approach are outside of the scope of this thesis and are addressed by the upcoming research work by Glen McCallum[2].

4.1.2.2 Integration with a Standard Information Model

The "curly braces" problem (see Section 2.1.1) is not unique to, but commonly associated with Arden Syntax. The shortcoming of the languages is caused partially by the lack of language integration with a standard patient information model and the absence of a uniform query language. To overcome this issue, we have proposed coupling of Arden with the CDA model (see Section 2.2.1.1) that can be referenced from Arden guidelines with XQuery language [130].

Figure 8 shows how a data slot statement from the MLM illustrated in Section 2.1.1 can be rewritten using CDA and XQuery.

---

[2] Netlab group, Department of Computer Science, University of Victoria.

```
raw_potassium := read last 3 from {
<results> {
    let $labEntries :=
document("ExtendedDataSet.xml")/ClinicalDocument/component/LabsSection/LabsEntry2
    for $entryChoice in $labEntries
        where $entryChoice/Observation/code = "potassium code"
        return
            <labResult code="{$entryChoice/Observation/code}"> {
            attribute value ($entryChoice/Observation/value},
            attribute date  ($entryChoice/Observation/effectiveDate}
            }
            </labResult>
}
</results>
 } where they occurred within the past 3 months);
```

Figure 8: Curly braces solution example

CDA documents can enclose codified patient information that is obtained from an EMR. The documents have a standard-based predefined structure that allows a uniform query and interpretation of the patient data by the Arden guidelines. XQuery language was selected as a flexible method for querying XML-based CDA documents.

While this is not a universal solution, it is a functional approach to solving fundamental problems of Arden Syntax. The advantages of using the CDA standard is that is developed by a large and active standards body, has a good chance of persistence,

represents a modeling effort of diverse group of participants and belongs to the same standards body as Arden, namely HL7.

4.1.2.3 Integration with Standard Medical Ontologies

Another cause for Arden's "curly braces" problem is the absence of integration with standard medical ontologies. The lack of unified terminology by which concepts in a guideline can be uniquely identified is a significant obstacle for knowledge interoperability.

Our approach to solving this aspect of "curly braces" problem is to leverage the RIM or more specifically CDA as its derivative. We have selected CDA in order to be consistent as it is also our interoperable information model of choice.

CDA is general enough to accommodate medical concept models [131] as it includes such classes as Medication, Observation, Procedure, and even more generic class Act. These classes can encapsulate medical concepts and their attributes such as the source of the concepts (i.e. unique id for the medical ontology), route, dosage and frequency of medication administration, etc.

Such encoding of medical terminology allows automatic validation and interpretation of clinical meaning associated with the medical concepts. Therefore, this approach not only solves this element of "curly braces" problem, but also improves data integrity and makes conceptualization easier.

4.1.2.4 Structured Recommendations

The ability to generate structured recommendations is an important characteristic of a guideline language. Arden Syntax is one of the few guideline languages that address this

issue. As part of the language specification, Arden contains a document type definition (DTD) for an MLM output which is referred to as a "Structured Message". This definition of the "Structured Message" provides a high-level categorization of recommendation elements that an MLM can produce in a response to the condition specified in the MLM logic slot. A recommendation can contain an <instruction> element that denotes a recommended clinical action, and a <choice.list> element that encompasses available options appropriate for the instruction.

In addition, the "Structured Message" may contain a reference for the provided recommendation as well as arbitrary text pertaining to the MLM output.

While Arden Syntax allows some recommendation structuring, we have identified it to be sufficient for the initial versions of our CDSS framework for several reasons. First, there is no consensus among EMR vendors on the recommendation structure of guidelines. Second, structuring recommendations is an immature area of research that demands further work. Therefore, we have adopted Arden's recommendations structuring standard as is.

4.2 Information Interoperability Approach

4.2.1 Clinical Document Architecture

To facilitate the interoperability of the patient-centric data transferred between the CDSS and EMRs, we have selected HL7 RIM as a meta-model and its derivative CDA model (see Section 2.2.1.2).

As all other HL7 specifications, CDA is an open public domain standard. Is continuously evolving CDA and is open to the inputs from CDA adopters through a member balloting process.

Therefore, CDA is consistent with our requirements for an open content and standard status of an information model. Next we show how our implementation of CDA satisfies other information interoperability requirements defined in Chapter 3.

4.2.1.1 Defined Structure

The CDA documents derive their structure, semantics, and data types form the RIM. CDA also makes use of the standard vocabulary domains, which are coded value sets. Some CDA vocabularies are HL7-defined; others can be drawn from external coding systems such as standard medical ontologies.

The CDA model consists of classes such as ClinicalDocument, Author, Organization, Patient, Observation, SubstanceAdministration, Procedure, Encounter, LabeledDrug, etc. These classes denote patient-specific information entities that are organized as document header and body.

The header communicates the context in which the document was created [29]. The purpose of the header is to facilitate clinical document exchange and management. It consists of four logical components:

- *Document Information* uniquely identifies the document and its relations to other documents, as well as conveys the document's confidentiality status.
- *Encounter Data* describes the setting in which the document originated.

- *Service Actors* are participants of the service that the document describes. Some examples of service actors include a care giver, a document authenticator, a transcriptionist, etc.
- *Service Targets* characterize patients or other significant participants.

A CDA body represents the actual content of the document – the clinical report. It can be either comprised of non-XML arbitrary text or structured sections and entries. CDA standard envisions three levels of specification which enforce different degrees of document structure. Level One represents a detailed header and a largely unstructured document body. Level Two allows layering of constraints, or templates, on top of Level One for prescribing section-level constraints for different document types. Level Three requires that documents are fully structured and constrainted on the level of sections and entries. Currently, HL7 is in the process of developing Level Two specification. Figure 9 shows a truncated example of a Level Two CDA document and its major components.

In our work we are concerned with interchanging computer-interpretable codified patient data as well as semi-structured guideline recommendations intended for presentation to the clinicians. Therefore, we have implemented our CDA document templates at Level Two as described in the next sections.

```
<ClinicalDocument>
    <-- CDA Header -->
    <Author>
    ...
    </Author>
    ...

    <StructuredBody>
        <ActiveProblemsSection>
            <Observation>
            ...
            </Observation>
        </ActiveProblemsSection>

        <SurgicalHistorySection>
            <Procedure>
            ...
            </Procedure>
        </SurgicalHistorySection>

    </StructuredBody>

</ClinicalDocument>
```

Figure 9: CDA Level Two document structure

4.2.1.2 Content Specification Process

"Interoperability by design" is the major goal of CDA standard. It provides a framework for representing various types of clinical documents and therefore is a broad specification. On the implementation level, however, the richness and flexibility of CDA must be constrained to define content specific data models which are referred to as CDA templates.

While HL7 CDA templates specification is in a draft state, there is no universally guaranteed process to derive or validate conformance against CDA templates. However, as suggested in the unofficial release of CDA Level Two Release Two document, there are several workable ways how CDA can be constrained today. We have reviewed different options for deriving templates such as OWL [113], DTD [132], RELAX NG [133], and W3C Schema [94]. In fact, the CDA standard has no

prohibitions for using either technology as long as the derived templates conform to the base CDA model. In our approach to defining CDA constraints, we have selected W3C Schema for the following reasons:

- The standard base CDA schema is specified in W3C Schema language, therefore no additional migration effort is required.
- W3C Schema provides a mechanism for creating a derived schema whose content model is a subset of the base schema. This method is referred to as restriction of schema types and can be validated by most XML parsers.
- There is a wide availability of tools for working with W3C Schema, unlike for other alternative technologies mentioned above.
- HL7 has successfully used restriction mechanism in its other specifications, for example for derivation of HL7 data types.

4.2.1.3 W3C Schema Restriction

The classes of the base CDA model are expressed in terms of complex types on the schema level. The CDA templates also may contain many of the same complex types, however with certain imposed constraints and restrictions. Derivation of complex types by restriction is a multifaceted feature of W3C Schema that is particularly useful in the case of template creation, where secondary types need to conform to the base primary type, but can also add their own constraints which go beyond those of the primary type. This implies that an instance of the template is also a valid instance of the base CDA model. Examples of acceptable restrictions on the CDA model include:

- Limiting the level of nesting

- Constraining vocabularies
- Enforcing sequences of types
- Changing the occurrence range for elements
- Making an optional attribute to being required

When using W3C Schema restriction mechanism, it is also important to be aware of its shortcomings. For example, derivation by restriction does not straightforwardly map to Object Oriented programming or relational databases because the content model of the complex type must be duplicated and refined. Additionally, the XML elements of the derived type have to be specified explicitly and manually, however all XML attributes are inherited implicitly and have to be ruled out manually.

4.2.2 CDA Level Two Templates

Using the restriction mechanism described above, we have developed a set of templates for encoding and exchanging interoperable clinical information. Four types of document templates are discussed in the following subsections. The content structure of these document templates was designed by Dr. Morgan Price[3], the medical expert associated with this thesis work. Furthermore, complete schemas and descriptions for the templates can be obtained from the thesis author on request.

4.2.2.1 Core Data Set

The "Core Data Set" document contains a subset of patient data that is initially sent from an EMR to the CDSS in response to a clinical event, such as a patient encounter. This document is defined as a minimum collection of patient data to determine which

[3] Morgan Price, MD, CCFP. University of British Columbia, Department of Family Practice.

preventive care guidelines apply to a given patient. The "Core Data Set" is essentially a cumulative patient profile as typically recorded in a patient's record – traditionally on the inside cover of the patient's paper chart.

The "Core Data Set" includes the following clinical components:

- Patient Demographics
    - Date of birth
    - Gender
    - Marital Status
- Active Medical Problems
    - Diagnoses for current medical problems
- Past Medical History
    - Diagnoses for previous significant conditions that may be episodic, in remission, or have been resolved
- Surgical History
    - Previous surgeries performed
- Medications
    - A list of patient's medications as recorded in the EMR
- Allergies and Adverse Reactions
    - Medication Allergies
    - Food Allergies
    - Environmental Allergies
- Family History
    - Significant diagnoses in the patient's family history

- Immunizations
    - Vaccinations against preventable diseases
- Social History
    - Significant conditions such as smoking, poverty, addictions, etc

4.2.2.2 Additional Data Request

In cases where additional patient data is required by the CDSS, an "Additional Data Request" document is used for formulating a query sent from the CDSS to the EMR. By the nature of preventive care guidelines selected for our CDSS, the information requests relate only to the lab-specific data that may be required as an input for some guidelines.

The body of the "Additional Data Request" document contains one structured section with a variable number of Observation classes which represent required laboratory results. Each Observation has a code element that contains a value from the LOINC [32] vocabulary that codifies the specific types of required laboratory data. In this document template, the Observation classes also contain other elements with empty values for the actual lab results and relevant date and time variables. The empty values, consequently, must be filled by the EMR and returned back to the CDSS.

Such approach to the information requests allows for an interoperable and technology-independent querying of heterogeneous EMRs.

4.2.2.3 Extended Data Set

The "Extended Data Set" is a template for a document that is sent from an EMR to the CDSS in response to the "Request for Additional Data". Note that an instance of this document would contain not only requested lab data, but also the entire content of the

original "Core Data Set" sent by the EMR initially for the same patient. This design decision is motivated by the intention to relieve the CDSS from the burden of cashing or persistently storing any patient data. The "Extended Data Set" is treated by the CDSS as an independent input document that is individually interpreted by the CDSS inference engine.

4.2.2.4 Recommendations

This document template provides a container for the end result of CDSS processing, namely patient-specific clinical recommendations and guidance. As with other templates defined here, this document contains a fully structured and codified header for providing document context and management. However, the body of the document is semi-structured with one section for holding a variable number of recommendations. The individual recommendations are represented as entries of narrative text which is unstructured in CDA terms. The CDA model does not explicitly provide constructs for structuring guideline recommendations. However, the text for recommendations can be marked-up according to other specifications such as Structured Message in Arden Syntax (see Section 2.1.1).

4.2.3 Leveraging Existing Tools

XML Special Interest Group was formed by HL7 organization in order to support the mission of using XML to make all HL7 specifications vendor- and system-independent. Consequently, CDA leverages XML and W3C XML Schema standards. Therefore CDA can be interpreted by XML-aware tools such as web browsers as well as various modeling and validation applications.

Within our work, we have effectively used hyperModel [134] Eclipse plug-in for graphically rendering and analyzing CDA schemas as UML models. hyperModel is capable of handling large and complex data models and can facilitate round-trip conversion between UML models stored in "xmi" format and W3C Schemas in "xsd" format. Unlike other tools with attempt to provide the similar functionality, hyperModel preserves important Schema attributes after model conversion into UML.

In addition, we leveraged various XML modeling tools for creating CDA templates and validating them against the base CDA schema.

4.3 Process Interoperability Approach

The CDSS process integration may be facilitated to different degrees: from integration into interactive and highly complex EMR workflows to minimal data exchanges between the CDSS and an EMR. Both approaches represent two sides of the spectrum: an ultimate vision of process integration and a limited, but practical solution.

When designing a communication interface for the process interoperability between our CDSS and various EMRs, we have met with a number of EMR vendors and consequently concluded the following. While EMR vendors welcome the possibility of enriching their systems with decision support functionalities, they also indicate that extensive changes of their systems for complex process integration with decision support technologies are not acceptable because the field of automated decision support is not sufficiently mature today.

Taking these factors into consideration, we attempted to strike a balance between the two sides of the spectrum by offering process interoperability that is effective, yet

flexible and not overly integration-intensive. With this motivation we selected a middleware technology and designed a four-step protocol for the process interoperability.

4.3.1 Middleware

From the review of various middleware technologies (see Section 2.5) we have selected Web Service as a platform for implementing service-oriented middleware. Although both CORBA and Web Services satisfy the requirement of platform independence, they differ by the following criteria.

4.3.1.1 Scalability

Scalability is important when the use load for a system is anticipated to grow over time.

In CORBA all communications between distributed systems have to take place via an Object Request Broker (ORB). Unless multiple ORBs are explicitly interconnected, this can be a potential bottleneck. Web Services, on the other hand, have a greater scalability in this aspect because they can communicate over the network without a dedicated intermediary.

4.3.1.2 Firewall Traversal

The reason why Web Services are generally considered friendlier to firewalls than CORBA is because SOAP connections take place over HTTP which uses port 80 as a standart. CORBA's IIOP protocol runs over TCP/IP and has no standard port. Where firewalls do not support IIOP traffic, it can be tunneled through HTTP via IIOP firewall products [135].

4.3.1.3 Vendor Independence

OMG is a group has attempted to define a standard for how ORBs should be built. However, these specifications allow different vendors to build their own "flavors" of CORBA [136]. So far CORBA vendors have been competitors who have no incentives to make their products interoperable. As a result, CORBA implementations from the same vendor are typically highly interoperable, but not necessarily so if they come from different vendors. In practice, to successfully interoperate every object in the distributed application has to run on the ORB from the same vendor. There are some cases where ORBs from different vendors do interoperate, but the interoperability typically does not extend into higher-level services such as transaction and security management.

With Web Service vendors, on the other hand, we see a higher degree of collaboration towards compatible implementations of the standards. This is an economically influenced cultural change from the competitive battles of the CORBA vendors in the 1990's.

In fact, a good evidence for the growing vendor conformance to the Web Service standards is the Web Services Interoperability Organization (WS-I) [137]. WS-I promotes Web Services interoperability across platforms, applications, and programming languages. It does so by providing the profiles which consist of interoperable specifications from World Wide Web Consortium (W3C) [94] and UDDI.org [106], conformance testing tools, and implementation guidelines.

4.3.1.4 Leveraging the Web

While CORBA extends itself to include the Web, Web Services represent its continued evolution. CORBA has proven itself robust and effective in the closed environments such as company intranets. Web Services, on the other hand, are suitable for more coarse-grained system integration across the Internet. This is partially due to the exchange of technology-neutral XML documents among Web Services, as well as growing number of compatible Web-centric standards [138]. Some examples of such high level languages and protocols leveraged by the Web Services are: description languages, semantic ontology mark-ups, security and encryption standards, business workflow and transaction protocols (see Section 2.5.1.4).

4.3.1.5 Transaction Management

Web Service technologies provide a set of interoperability constructs that enable collaborations between the distributed systems. However, currently Web Services by default use Web Service Description Language (WSDL), which can only provide information about Web Service data types, methods, message format, protocol, and the Universal Resource Identifier (URI). Such interface description is not sufficient for the stand-alone CDSS that must engage in a series of predefined communication acts with other systems. These communication protocols, or multi-step data exchanges, correspond to the concept of distributed transactions.

We have extended the Web Service interface with Business Process Execution Language for Web Services (BPEL4WS) to include the definition of valid transactions to ensure consistent communication patterns and desired outcomes. BPEL4WS specifies

the expected document exchange behavior of the two parties involved in the protocol: the CDSS and an EMR. The incoming and outgoing communications of the CDSS are managed by a BPEL4WS transaction mechanism which provides fault handling in the application specific manner. Although BPEL4WS can also facilitate long-running transactions, compensations, and complex service co-ordinations via extensions for WS-Transaction [139] and WS-Coordination [140], these features are out of scope for our work in process integration. The communication protocol for governing the CDSS transactions is described next.

4.3.1.6 Communication Protocol

In order to fulfill the requirements for process interoperability, we have designed a communication protocol that defines the level of integration between the CDSS and EMRs. The exchangeable information units are CDA documents of the four types defined in Section 4.2.2.

To allow a high level of communication granularity for the EMRs, we have defined a four-step protocol for document exchanges, as demonstrated in Figure 10.

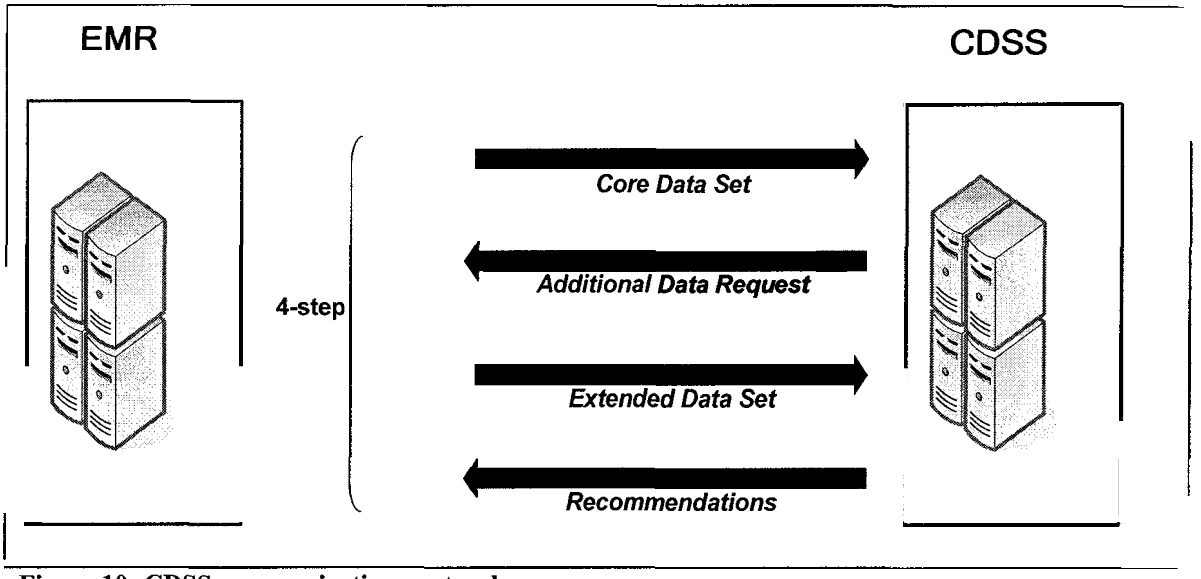

Figure 10: CDSS communication protocol

In the four-step document exchange, an EMR initially only sends a "Core Data Set" document (see Section 4.2.2.1) which contains predefined types of patient data. If the CDSS requires additional data such as patient's laboratory results, an EMR provides the "Extended Data Set" and only then obtains CDSS patient-specific recommendations. It is intentionally designed that an EMR always gets the "Recommendations" document as the outcome. This document is returned to the EMR regardless of whether any actual recommendations apply to the patient. This ensures the acknowledgement by both systems that the protocol was executed successfully.

The communication protocol is exposed on the CDSS interface via the BPEL4WS (see Section 2.5.1.4.6).

Chapter 5 Conceptual Architecture

This chapter outlines the conceptual architecture of the CDSS followed by the detailed component interface specifications.

5.1 Component Architecture

This component-based architecture, illustrated in Figure 11, allows for a plug-and-play framework that is extensible, reconfigurable, and evolvable.

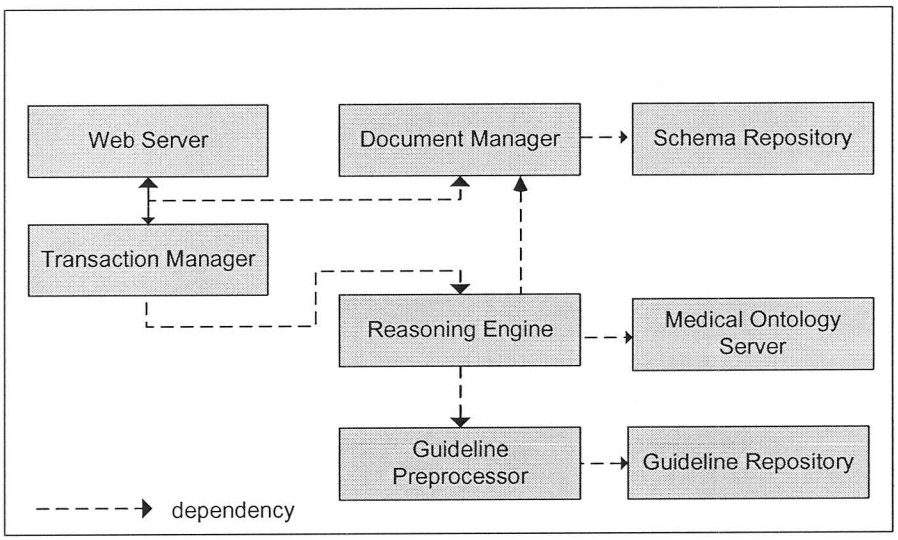

Figure 11: Conceptual component architecture

5.1.1 Web Server

The Web Server controls communications with the EMRs which access the stand-alone CDSS. The Web Server handles the following tasks:

- *Exposing vendor-agnostic communication interface*

The Web Server publishes CDSS service interface on the Web. Such approach provides a level of abstraction between the exposed interface and the actual implementation of the CDSS. This separation of concerns has the following advantages. First, the underlying implementation can evolve without necessarily requiring any changes to the interface. Second, a system that wants to communicate with the CDSS needs to know only the standard interface definition, thus avoiding the complexity and technology specificity of the underlying CDSS. The interface also describes the document-flow oriented communication protocol (see Section 4.3.1.6) that defined the allowed interactions with the CDSS.

- *Ensuring security*

The Web Server also addresses security aspects of CDSS connectivity by employing XML-based security mechanisms for solving problems related to authentication, role-based access control, distributed security policy enforcement, and message layer security.

5.1.2 Transaction Manager

Besides defining protocols for transactions on the interface level, a CDSS must also have the capability to assure their validity and integrity on the application level. This prerequisite for transaction validation and reliable coordination of operations across Web services is addressed by the Transaction Manager component. The Transaction Manager implements BPEL4WS and has the ability to receive incoming documents as well as calling other Web Services. This component ensures that the participants of the distributed transaction (the CDSS and EMRs) interact according to the predefined

communication protocol and achieve a mutually agreed outcome such as generation of clinical recommendations based on provided patient data.

The Transaction Manager can be seen as a gatekeeper between the Web Server which accepts incoming documents and the CDSS reasoning engine. The Transaction manager allows the CDSS engine invocation only if the incoming documents are valid and constitute parts of the allowed transactions.

This separation of concerns also allows the reasoning engine to remain independent of any business rules related to the communication with the CDSS.

It is important to note however, that in the proposed CDSS architecture and communication protocol, long-running transactions, roll-backs, and compensation functionalities are out of scope, although they can be accommodated by BPEL4WS [141].

5.1.3 Document Manager

The Document Manager handles incoming and outgoing CDA documents and has two primary functions:

- *Document Validation*

  This feature involves validation of the incoming and outgoing CDA documents against predefined templates (see Section 4.2.2). The structural validation involves comparing a given CDA document instance with the template schema associated with this document.

- *Document Conversion*

  In a document-based exchange paradigm, it is also necessary to facilitate transformation between the incoming CDA documents and the data representation

model required by the reasoning engine, and vise versa. This process is governed by the predefined transformation schemas and is specific to the document format and reasoning engine of choice. In our framework, we are leveraging the C Language Integrated Production System (CLIPS) (see Section 5.1.8) as a reasoning engine; therefore, the bidirectional document conversion takes place between the CDA and native CLIPS formats.

5.1.4 Schema Repository

The Schema Repository is a library that contains xml schemas for validation and transformations of CDA documents performed by the Document Manager component. Each schema is indexed and catalogued according to its properties such as type (e.g. document schema or transformation schema), version information, and unique identifiers.

5.1.5 Medical Ontology Server

The medical terminology used in the guidelines and patient profiles must be semantically interpreted by the reasoning engine. For example, if a patient's profile includes medications with brand names and the CDSS guidelines use generic medication names, then semantic mapping is required to establish a relation between the terminologies for medications.

This semantic mapping process involves retrieval and interpretation of codes from either locally or remotely stored medical ontologies. Local storage of such ontologies may not a feasible approach for two reasons. First, medical ontologies may change and evolve frequently. Second, they are often prohibitively large to be managed by individual CDSSs. Therefore, semantic mapping of coded data between guidelines and

patient data should be performed by means of referencing designated medical authoritative bodies that specifically manage medical ontologies. One of the largest and most mature entities that provide this type of functionality is the United States National Library of Medicine which designed a UMLS Knowledge Source framework as a collection and cross-reference for medical ontologies. The Knowledge Server can be programmatically accessed and used in the CDSS with consideration for individual licenses of the medical ontologies that constitute UMLS.

5.1.6 Guideline Repository

The Reasoning Engine operates on clinical practice guidelines that are stored in the Guideline Repository as algorithmic representations of knowledge. In our CDSS framework, the Guideline Repository store guidelines encoded in Arden Syntax. However, because guidelines are preprocessed by an intermediary component before they are consumed by the Reasoning Engine, multiple guideline formats and repositories can potentially be accommodated by having appropriate Guideline Preprocessors.

5.1.7 Guideline Preprocessor

The Guideline Preprocessor converts clinical practice guidelines from their encoding language to the native data format of the Reasoning Engine. In our framework, the Preprocessor performs dynamic unidirectional transformations from Arden Syntax guidelines to the CLIPS data model. Other types of Preprocessors can be added as the system evolves, without affecting its other components.

5.1.8 Reasoning Engine

The Reasoning Engine is the core CDSS component. It operates on decision algorithms, medical knowledge, and specific patient data in order to generate recommendations and guidance for the clinician.

We have selected an existing expert system CLIPS [142] as our Reasoning Engine. CLIPS is an open source product that has been developed by NASA in 1985 and used in many applications with automated reasoning. It is not a domain specific system and can operate on various types of knowledge, including preprocessed Arden Syntax guidelines and patient information. The inputs and outputs of CLIPS are represented in a standardized CLIPS format.

5.2 Component Interface Specifications

Figure 12 shows the component diagram for our CDSS where dependencies are identified on the interface level.

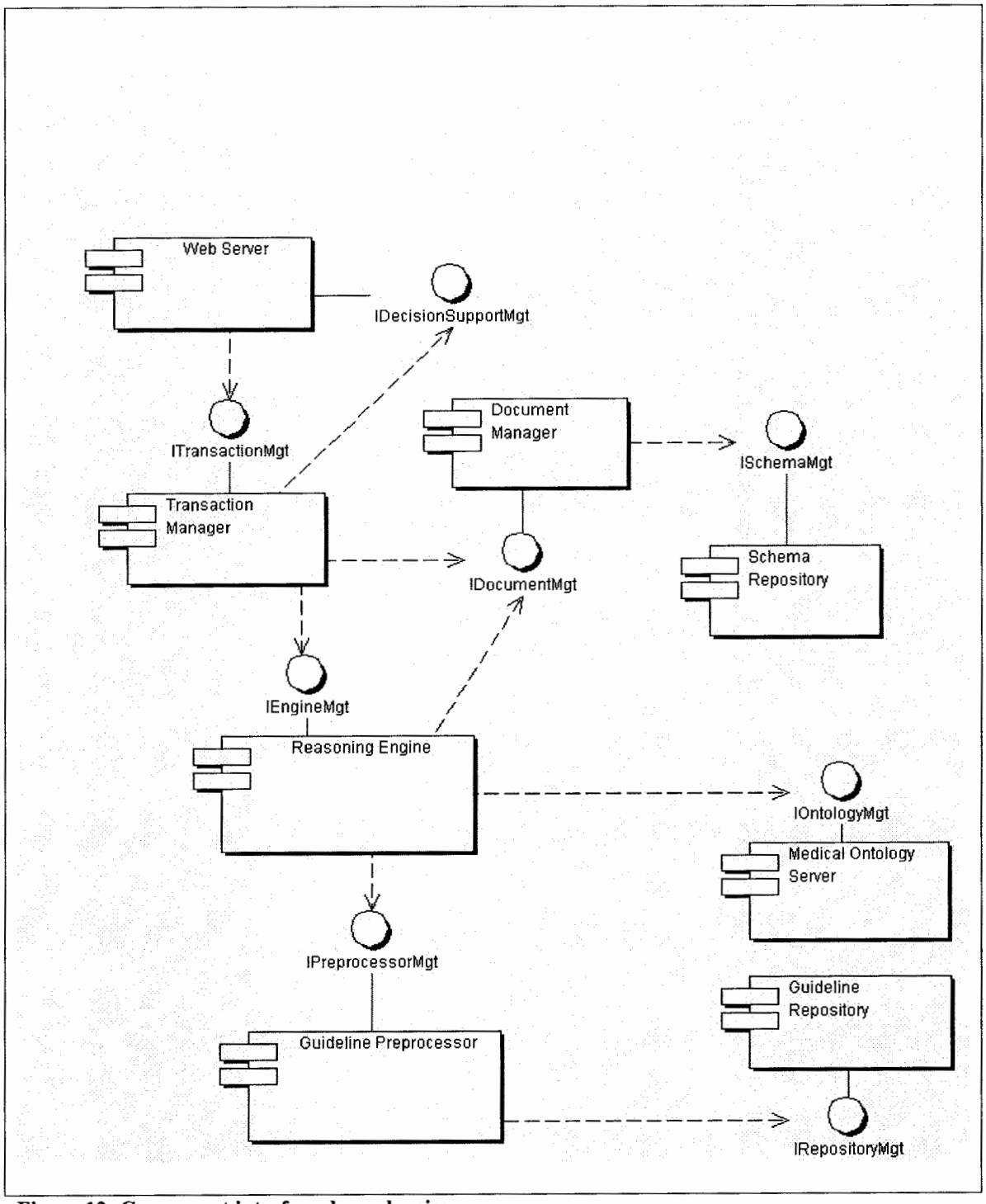

Figure 12: Component interface dependencies

Next we specify the interfaces and their methods in detail in order to better describe what functionality each component offers.

IDecisionSupportMgt	
Methods	**Method Description**
acceptCoreDataSet (in CoreDataSet: CDAdoc, out AdditionalDataRequest: CDAdoc)	Accepts a CDA document of type "Core Data Set" and returns a request for missing patient data in the form of another CDA document of type "Additional Data Request"
acceptExtendedDataSet (in ExtendedDataSet: CDAdoc, out Recommendations: CDAdoc)	Accepts a CDA document of type "Extended Data Set", which includes the a patient profile with the requested laboratory or physical examination information. Returns patient-specific clinical recommendations in the form of another CDA document of type "Recommendations"
sendDocumentToEMR (in document: CDAdoc, in destination: URL)	Sends a CDA document from the CDSS to an EMR identified by a URL. The transaction Manager invokes this method.

ITransactionMgt	
Methods	**Method Description**
isTransactionValid (in documentId: int, out result: bool)	Checks whether an incoming CDA document is part of the valid transaction (according to the predefined communication protocol). The method determines the transaction validity based on the document id and its conformance to the appropriate CDA template schema (this check is performed by the Document Manager). In case of the document id being null, the Transaction Manager may assign a new transaction id. The method returns true if the transaction is valid, otherwise returns false.

IDocumentMgt

Methods	Method Description
isDocumentValid (in document: CDAdoc, out result: bool)	Checks whether the given CDA document is valid by validating it against an appropriate schema for the given document type. Returns true if the document is valid, otherwise returns false.
convertToEngineFormat (in document: CDAdoc, out engineFile: File)	Converts a CDA document into the native format of the Reasoning Engine. Returns a file that can be directly used as the input for the Reasoning Engine.
convertFromEngineFormat (in engineFile: File, out document: CDAdoc)	Converts the output from the Reasoning Engine into a CDA document of an appropriate type: either "Recommendations" or "Additional Data Request".

ISchemaMgt	
Methods	**Method Description**
getTemplateSchema (in templateId: int, out schema: Schema)	Retrieves a CDA template schema based on the provided template id.
getTransformationScriptt (in templateId: int, out script: Script)	Retrieves a transformation schema for converting a CDA document into the input file for the Reasoning Engine. An appropriate transformation script is selected based on the template id of the CDA document.

IEngineMgt	
Methods	**Method Description**
processInformation (in engineFile: File, out engineFile: File)	Processes patient information and returns a result of the inferencing process. Both input and output are files in the required format of the Reasoning Engine.

IOntologyMgt	
Methods	**Method Description**
isClassTheSame (in concept1: String, in concept2: String, out result: bool)	Compares two medical concepts and determines if they belong to the same class (pharmacologic or therapeutic drug class; disease group, etc). Returns true if concept class is the same, otherwise returns false.
getClass (in concept: String, out class: int)	Determines a classification class that a concept belongs to (pharmacologic or therapeutic drug class; disease group, etc). Returns the id for the class.

IPreprocessorMgt	
Methods	**Method Description**
getGuidelines (in criteria: LogicalExpression, out engineFile: File)	Retrieves clinical practice guidelines that are preprocessed for the consumption by the Reasoning Engine. The guidelines are selected on the basis of an inclusion criteria, for example "all preventive care guidelines", etc. The inclusion criteria is formulated as a logic expression.

IRepositoryMgt	
Methods	**Method Description**
getGuidelines (in criteria: LogicalExpression, out preprocessorFile: File)	Retrieves clinical practice guidelines from the Knowledge Base. The guidelines are selected on the basis of inclusion criteria, for example "all preventive care guidelines", etc. The inclusion criteria are formulated as a logic expression. The guidelines are returned in their original encoding format, such as Arden Syntax.

Chapter 6 Discussion

This chapter presents a discussion on several issues that were identified in the process of the CDSS design. We categorize them according to the three main topics of our research: knowledge, information, and process interoperability. We discuss both the evaluation of our approach and the challenges associated with the CDSS interoperability.

6.1 Knowledge Interoperability

6.1.1 Arden Syntax and Formal Semantics

While syntax defines the structure of well-formed models, semantics assigns meaning to the models and ensure their unambiguous interpretation. The importance of formal semantics has been especially emphasized in the computer science literature [143]. However, formalization is equally important for knowledge representation languages which must be unambiguous, computer interpretable, and interoperable.
Other advantages of formalized and standardized semantics are that tools can be developed by a software engineering community and leveraged for building knowledge-based systems. Some examples of such tools are semantic validators and compilers for the language.

Arden Syntax, unfortunately, does not have formal semantics defined as part of its standard. There is no machine processible mathematical model that defines the meaning of the language constructs. Although the Arden specification is well described, the meaning of language constructs is communicated only through structured English. This

approach is seemingly clear, but yet allows the specification to be interpreted differently by different readers as well as implementers of Arden Syntax reasoning engines. The optimal solution for this problem, of course, would be writing a formal semantic definition for Arden. For example, similarly to the PROforma guideline language, operational semantics (see Section 2.1.3) could be developed, where interpreting automation [144] allows expressing the modeling language by providing a mechanism that determines the effect of any language operation. However, the development of a semantic formalism for a language is out of scope for this thesis because of time and resources constraints. Instead, the priority is given to the interoperability aspects of a CDSS.

For these reasons, we proposed a practical approach to defining Arden Syntax semantics by developing a reference implementation of an Arden execution engine based on the CLIPS expert system [142].

This is somewhat consistent with a technique for defining translational semantics (see Section 3.1.2) because we compile Arden into CLIPS language. However, we do not formally specify this mapping other than by means of open sourcing our reference implementation of the Arden Syntax execution engine. The actual specification of rules that define how Arden can be mapped into CLIPS is left to the knowledge modeling community for future work.

6.1.2 Curly Braces Problem

The "curly braces" problem (see Section 2.1.1) is a well known issue associated with the implementation of Arden Syntax. It stems from the original application of Arden for

embedding MLMs into proprietary clinical systems. In fact, the initial version of Arden was based on an internal decision support scheme of a proprietary EMR called HELP. This limitation of Arden Syntax has been discussed in several publications [131]. There is a consensus of many clinical domain experts that the solution to this problem lies in leveraging a standard patient information model such as HL7 RIM [129]. Consistent with this opinion, we have selected the CDA standard as a RIM derivative for relating Arden-encoded clinical knowledge to patient data and medical concepts.
This choice is motivated by two factors. First, RIM itself is a high-level and large meta-model that needs to be refined for a specific domain and clinical purpose. CDA allows such specialization of RIM by means of template definition for CDA document types (see Section 4.2.2). Second, we have adopted the CDA standard for the interchange of clinical documents between the CDSS and EMRs. Therefore, it is efficient to perform direct mapping of patient data and medical concepts from the incoming CDA documents to the Arden MLMs.

In fact, this approach is comparable with the GLIF framework, where another derivative of the RIM, Uniform Service Action Model (USAM) [145], is used for referencing external data for the guidelines.

6.1.3 Arden Syntax and Complex Guidelines

Our selection of Arden Syntax was primarily driven by the fact that it is the only standardized and open guideline modeling language available in clinical academia and industry. At the same time, we are well aware of several well-known Arden limitations. While in our approach we solve the "curly braces" problem (see Section 4.1.2), an important issue remains in regards to modeling multi-step complex guidelines in Arden.

Although technically MLMs can be chained to model multi-step guidelines, there is no specification how it should be done in a standard fashion. For example, the current version of Arden Syntax does not allow for a clear definition of dependencies among different MLMs. Therefore, such guideline modeling constructs as branching and synchronization of guideline steps would be difficult to accommodate.

Meanwhile, this is not a prohibitive feature for our CDSS because we have implemented simple guidelines for preventive care and reminders that are each encapsulated in a single MLM. However, this aspect of Arden should be considered for the future evolution of our CDSS guideline knowledge base.

There are several options that can be resorted to for modeling more complex guidelines. First, the modular architecture of the proposed CDSS would allow for adoption of another guideline formalism that can facilitate more advanced guideline constructs. One of such languages may be PROforma (see Section 2.1.3) if it officially becomes an open standard.

Second, the future work for this thesis project may include participation in the evolution process for Arden Syntax that would address formal linking between MLMs. This could be complimentary to the HL7 efforts that are underway in the same direction [129].

6.2 Information Interoperability

6.2.1 Clinical Document Architecture

The advantage of using the Clinical Document Architecture (CDA) is that it belongs to a group of standards fostered by HL7. HL7 is a large international standard organization

aimed at developing and maintaining information standards for systems interoperability. Therefore such clinical information models as RIM and its derivative, including CDA, have a potential for wide acceptance, persistence, and sustainability. Additionally, the design of CDA leverages modeling efforts by a diverse group of HL7 participants.

On the other hand, the disadvantage of using CDA is that it is a "moving target", a new standard anticipated to evolve and change in the future. Currently, there are few early adopter of CDA and therefore are few reference implementations and CDA-specific tools exist.

Moreover, the CDA standard has been defined for the first of its three envisioned levels (see Section 2.2.1.2). CDA Level One only sets requirements for the structure of the CDA document header where primarily administrative information is encoded. The structure of the CDA document body is left undefined. This level of document structuring is not sufficient for automated document processing where structure and semantics must be clearly predefined in the form of document templates. This problem is well recognized and will be addressed by CDA Level Two and Level Tree standards. However, the latter two are underway and have not been yet released. Therefore, the CDA developers are left with no consensus on how a CDA template should be derived. Several alternatives have been explored to address this issue in the development community. Among them are OWL [113], DTD [132], RELAX NG [133], and W3C Schema [94]. We have adopted an approach of using the latter one to specify CDA templates. W3C Schema offers such advantages such as providing a mechanism for creating a derived schema whose content model is a subset of the base schema. This method is referred to as restriction of schema types (see Section 4.2.1.3) and can be

validated by many XML modeling tools. Moreover, HL7 has also used the schema restriction mechanism in its other specifications, for example for derivation of HL7 data types. Therefore it is likely that subsequent CDA standards may also adopt W3C Schema approach for document templating. However, this remains to be seen.

6.3 Process Interoperability

6.3.1 Communication Protocol

The minimal communication process between the stand-alone CDSS and an EMR would consist of a two-step document interchange, where a patient profile is sent to the CDSS in return for the patient-specific clinical recommendations. However, the problem with this approach is that it is difficult to predefine what patient information should be provided to the CDSS as a single document input.

Although we have defined a "Core Data Set" template as a basic patient profile (see Section 4.2.2.1), many clinical guidelines typically also require additional information such as laboratory or physical examination results.

The difficulty arises because an EMR may not "know" what patient information maybe relevant unless it is performing its own internal decision support. Thus the decoupling of an EMR and a CDSS requires that the design of the process interoperability take these issues into careful consideration.

We have evaluated several approaches for the process interoperability. First, if a two-step protocol is used, a CDSS may define the content of the input document according to the clinical guidelines that the CDSS operates on. For example, if a CDSS only performs decision support for colon cancer reminders, then it is possible to specify that the input

patient profile must include lab results for occult blood test. However, introducing the dependency between the CDSS knowledge base and its process interface is not an appropriate solution if the knowledge content is large and is anticipated to evolve. Second, a more generic approach may be taken by requiring that the Core Data Set also includes the laboratory and physical exam information for the specific period of time. Introducing such temporal cut off for the patient information sent from an EMR may result in the interchange of unnecessary information, or omission of important information for CDSS processing.

After reviewing the above options, we have adopted yet another approach that is more flexible and efficient. We introduced a four-step communication protocol which allows the CDSS to process the Core Data Set and request specific additional data from an EMR if needed (see Section 4.3.16).

This four-step protocol entails an additional level of complexity on both the EMR and CDSS sides. An EMR is required to be able not only to assemble a basic patient profile (the "Core Data Set"), but also to provide additional patient data on demand. The CDSS, on the other hand, must be able to determine missing data from the combinatory analysis of the patient profile and applicable clinical guidelines. Although the four-step protocol requires more implementation effort than the two-step protocol, it is more efficient and focused on an interactive CDSS integration. In the future versions of the CDSS, this protocol can be taken further to introduce a more intelligent and context-specific decision support.

Chapter 7 Conclusion

7.1 Summary

In order to provide quality care, clinicians must be up-to-date with the latest knowledge in the medical field [146]. However, the amount of available clinical knowledge is increasing rapidly especially with the expansion of the Internet. It is estimated that the amount of clinical knowledge is going to increase fourfold during a professional lifetime of a modern clinician [147]. The easier access to information does not necessarily mean that available information is relevant for a particular patient case. The problem of information overload therefore becomes a significant barrier for effective and efficient clinical decision making [148]. Providing a service that can be integrated into a number of EMR systems and offer just-in-time relevant medical knowledge can lead to a significant improvement of the health care quality [149] [4].

Due to the interdisciplinary nature of this work, we have done an investigation into various health informatics fields as a prerequisite for CDSS design. We have researched topics related to guideline encoding and reasoning, patient information models, and interoperability mechanisms for distributed systems. Furthermore, we explored the related work in other decision support systems and discovered that the domain of clinical decision support is relatively immature and requires continuous development.

Driven by these motivations, this thesis introduced an open source stand-alone CDSS that can be integrated with other clinical systems in a distributed setting of care delivery. Our work primarily focused on the interoperability aspects such as knowledge, information, and CDSS processes. As a result, a component-based conceptual

architecture was defined for an interoperable stand-alone CDSS. In the process of the overall work a lot of interesting and challenging issues were identified in regards to the clinical decision support, as was presented in our discussion (see Chapter 6).

The major contribution of our work is the development a CDSS framework that is agnostic of EMRs and based on open source interoperability standards. The process interface exposed by the CDSS allows the integration of its functionality into various clinical contexts.

7.2 Future Work

The work presented in this thesis is a documentation of an ongoing collaborative project[4] which engages the expertise from the clinical and software engineering domains. Therefore, the new ideas that where generated in the process of this thesis writing will be carried over as future work for refining and enhancing the proposed CDSS framework. The future work will include both the research and practical implementation and will be aligned along the three main interoperability aspects emphasized in this thesis.

Knowledge Interoperability:

- *Accommodating complex guidelines:* As discussed in Section 6.1.3, Arden Syntax poses limitations on the types of guidelines it can model. More specifically, modeling and managing complex guidelines by chaining MLMs is problematic due to the limitations of the Arden language. The objectives of our future work are both to pursue more expressive guideline models like PROforma and to take an active participation in the evolution of Arden Syntax through collaborations with HL7.

EGADSS Project. http://egadss.sourceforge.net/

- *Structuring guideline recommendations:* Although Arden syntax provides a document type definition (DTD) for the MLM output (see Section 4.2.2.4), it defines a weak structure for the clinical recommendations. A more robust model is required to improve interoperability between clinical systems that exchange guideline-based knowledge. While the semantics and contents of the recommendations can be very diverse depending on the context of the clinical encounters, there are many similarities in how different guidelines and guideline languages organize their recommendation [26]. This area of knowledge management has gotten very little attention in the literature [125]. However, we clearly recognize its importance and intend to pursue it further.
- *Integrating a knowledge management environment into the CDSS*: Up to this point our work has largely focused on the interoperability functionalities of the CDSS. Another equally important aspect that must be addressed is the knowledge management interface for the CDSS. While low-level manual guideline encoding is acceptable at the initial stages of CDSS development, it becomes unfeasible as the knowledge base grows and requires adequate management mechanisms. We are specifically interested in investigating such knowledge management tools as Protégé [150].

Information Interoperability:

- *Refining the CDA templates:* As the CDA templating specification emerges and matures, we anticipate that our approach has to be revisited and made consistent with the upcoming CDA standards for Level Two and Level Three. It is also

likely that HL7 may create an official registry for CDA templates for various clinical domains including decision support.

Process Interoperability:

- *Further development of the CDSS process interoperability protocol*: The communication protocol proposed in this thesis intends to provide a practical solution for process interoperability while entailing minimal required changes for the EMRs. This is largely dictated by the fact that the decision support domain is not yet mature and thus EMR vendors are not willing to take a high risk of implementing many changes into their systems. We are hoping, however, that our work contributes to the CDSS community and serves as an incentive for the EMRs to produce patient data that is more interoperable and standard-conformant. We have already started working directly with several EMR vendors in Canada to move advanced and effective clinical decision support closer to reality.

Bibliography

[1] The International Institute of Management Development, *World Competitiveness Yearbook*. Switzerland, 2004.

[2] Canadian Institute for Health Information, *Health care in Canada*, 2003.

[3] W. A. Yasnoff, B. L. Humphreys, J. M. Overhage, D. E. Detmer, P. F. Brennan, and R. W. Morris, "A consensus action agenda for achieving the national health information infrastructure," *Journal of American Medical Informatics Association*, vol. 11, pp. 332-8, 2004.

[4] D. L. Hunt, R. B. Haynes, S. E. Hanna, and K. Smith, "Effects of computer-based clinical decision support systems on physician performance and patient outcomes: a systematic review," *Journal of American Medical Informatics Association*, vol. 280, pp. 1339-46, 1998.

[5] D. W. Bates, J. M. Teich, and J. Lee, "The impact of computerized physician order entry on medication error prevention," *Journal of American Medical Informatics Association*, vol. 6, pp. 313-21, 1999.

[6] D. W. Bates, M. Cohen, L. L. Leape, J. M. Overhage, M. M. Shabot, and T. Sheridan, "Reducing the frequency of errors in medicine using information technology," *Journal of American Medical Informatics Association*, vol. 8, pp. 299-308, 2001.

[7] S. Bakken, "An Informatics Infrastructure Is Essential for Evidence-based Practice," *Journal of American Medical Informatics Association*, vol. 8, pp. 199-201, 2001.

[8] M. Webster, *Merriam-Webster's Collegiate Dictionary*: Springfield: Merriam-Webster, 1998.

[9] D. W. Bates, M. Ebell, E. Gotlieb, J. Zapp, and H. C. Mullins, "A proposal for electronic medical records in U.S. primary care," *Journal of American Medical Informatics Association*, vol. 10, pp. 1-10, 2003.

[10] J. Cartwright, S. de Sylva, M. Glasgow, R. Rivard, and W. J, "Inaccessible information is useless information: addressing the knowledge gap," *Journal of Medical Practice Management*, vol. 18, pp. 36-41, 2002.

[11] M. E. Johnston, K. B. Langton, R. B. Haynes, and A. Mathieu, "Effects of computer-based clinical decision support systems on clinician performance and patient outcome. A critical appraisal of research.," *Annals of Internal Medicine*, vol. 120, pp. 135-42, 1994.

[12] Y. Shahar, Automated support to clinical guidelines and care plans: the intention-oriented view, OpenClinical, http://www.openclinical.org/docs/int/briefingpapers/shahar.pdf, August 4, 2004

[13] Institute of Medicine, *Clinical Practice Guidelines: Directions for a New Program*. Washington, DC: National Academy Press, 1990.

[14] D. Wang, M. Peleg, S. W. Tu, A. A. Boxwala, R. A. Greenes, V. L. Patel, and E. H. Shortliffe, "Representation primitives, process models and patient data in computer-interpretable clinical practice guidelines: a literature review of guideline representation models.," *Int J Med Inform*, 2002.

[15] M. Peleg, A. A. Boxwala, O. Ogunyemi, Q. Zeng, S. Tu, and R. Lacson, "GLIF3: the evolution of a guideline representation format," presented at American Medical Informatics Association Annual Fall Symposium, Los Angeles, CA, 2000.

[16] P. L. G. Hripcsak, T.A. Pruor, O.B. Wigertz, P.B. Clayton, "Rationale for the Arden Syntax," *Computers in Biomedical Research*, vol. 27, pp. 291-324, 1994.

[17] D. Wang, "Translating Arden MLMs into GLIF Guidelines – A Case Study of Hyperkalemia Patient Screening," presented at Symposium on Computerized Guidelines and Protocols, Prague, Czech Republic, 2004.

[18] G. Hripcsak, P. Ludemann, T. A. Pruor, O. B. Wigertz, and P. B. Clayton, "Rationale for the Arden Syntax," *Computers in Biomedical Research*, vol. 27, pp. 291-324, 1994.

[19] L. Ohno-Machado, J. H. Gennari, and S. N. Murphy, "The guideline interchange format: a model for representing guidelines," *Journal of American Medical Informatics Association*, vol. 5, pp. 357-372, 1998.

[20] E. Pattison-Gordon, "Object Data Interchange Format (ODIF)," Decision Systems Group. Brigham and Women's Hospital, Boston, Massachusetts DSG-96-04, 1996.

[21] M. Sordo, O. Ogunyemi, A. A. Boxwala, R. A. Greenes, and S. Tu, "GELLO: An Object-Oriented Query and Expression Language for Clinical Decision Support," Report prepared for OpenClinical 2003.

[22] "Design and implementation of the GLIF3 guideline execution engine," *Journal of Biomedical Informatics*, vol. 37, pp. 305-18, 2004.

[23] J. Fox, N. Johns, C. Lyons, A. Rahmanzadeh, R. Thomson, and P. Wilson, "PROforma: a general technology for clinical decision support systems.," *Computer Methods & Programs in Biomedicine*, vol. 54, pp. 59-67, 1997.

[24] B. Potter, J. Sinclair, and D. Till, *An Introduction to Formal Specification and Z*: Prentice Hall, 1996.

[25] D. R. Sutton and J. Fox, "The Syntax and Semantics of the PROforma guideline modelling language.," *Journal of the American Medical Informatics Association*, vol. 10, pp. 433-443, 2003.

[26] M. Peleg, S. Tu, and J. Bury, "Comparing Computer-interpretable guideline models: a case-study approach.," *Journal of American Medical Informatics Association*, vol. 10, pp. 52-68, 2003.

[27] Health Level 7 Canada, http://secure.cihi.ca/cihiweb/dispPage.jsp?cw_page=infostand_hl7can_e., 2004, August 12

[28] International Standards Organisation, "Reference Model of Open Systems Interconnection," ISO/TC97/SC15 N227, 1979.

[29] R. H. Dolin, L. Alschuler, C. Beebe, P. V. Biron, S. L. Boyer, and D. Essin, "The HL7 Clinical Document Architecture," *American Medical Informatics Association*, vol. 8, pp. 552-569, 2001.

[30] D. A. Lindberg, B. L. Humphreys, and A. T. McCray, "The Unified Medical Language System," *Methods of Information in Medicine*, vol. 32, pp. 281-291, 1993.

[31] R. A. Cote, D. J. Rothwell, J. L. Palotay, R. S. Beckett, and L. Brochu, *The Systematized Nomenclature of Medicine: SNOMED International.* Northfield, IL: College of American Pathologists, 1993.

[32] S. M. Huff, R. A. Rocha, C. J. McDonald, G. J. de Moor, T. Fiers, and W. D. J. Bidgood, "Development of the Lgoical Observation Identifier Names and Codes (LOINC) vocabulary," *Journal of American Medical Informatics Association*, vol. 5, pp. 276-292, 1998.

[33] I. D. Adams, M. Chan, P. C. Clifford, W. M. Cooke, B. Dallos, and F. T. de Dombal, "Computer aided diagnosis of acute abdominal pain: a multicenter study," *British Medical Jounal*, vol. 293, pp. 800-804, 1986.

[34] F. T. de Dombal, D. J. Leaper, J. R. Staniland, A. P. McCann, and J. C. Horrocks, "Computer-aided diagnosis of acute abdominal pain," *British Medical Jounal*, vol. 1, pp. 376-380, 1972.

[35] M. A. Musen, Y. Shahar, and E. H. Shortliffe, "Clinical decision-support systems.," in *Medical Informatics: Computer Applications in Health Care and Biomedicine.*: Springer-Verlag, pp. 600-39, 2000.

[36] E. H. Shortliffe, *Computer-based medical consultations: MYCIN*. New York: Elsevier, 1976.

[37] E. H. Shortliffe, *Medical informatics Computer applications in healthcare and biomedicine*, Second ed. New York: Springer, 2001.

[38] J. McCarthy, "Some expert systems need common sense," in *Computer Culture: The Scientific, Intellectual and Social Impact of the Computer*, vol. 426, pp. 129-136, 1984.

[39] T. A. Pryor, R. M. Gardner, P. D. Clayton, and H. R. Warner, "The HELP system," *Journal of Medical Systems*, vol. 7, pp. 87-102, 1983.

[40] M. A. Musen, S. W. Tu, A. K. Das, and Y. Shahar, "EON: a component-based approach to automation of protocol-directed therapy," *Journal of the American Medical Informatics Association*, vol. 3, pp. 367-388, 1996.

[41] M. Musen, "Domain ontologies in software engineering: Use of PROTEGE with the EON architecture," *Methods of Information in Medicine*, vol. 37, pp. 540-550, 1998.

[42] P. Ram, D. Berg, S. Tu, J. G. Mansfield, Q. Ye, and R. Abarbanel, "Executing clinical practice guidelines using the SAGE execution engine," Stanford University SMI-2003-0971, 2003.

[43] D. S. Blank, L. A. Meeden, and J. B. Marshall, "Exploring the symbolic /subsymbolic continuum: a case study of RAAM," Department of Computer Science, Indiana University, Bloomington, Indiana: TR 332, 1991.

[44] K. Okagaki, "Using expert systems to validate and verify neural networks," presented at INNC-90, 1990.

[45] A. R. Golding and P. S. Rosenbloom, "Improving rule-based systems through case-based reasoning," presented at AAAI-91, Menlo Park, CA, 1991.

[46] J. Wnek and R. Michalski, "Comparing symbolic and subsymbolic learning: three studies," *Machine Learning: a multistrategy approach*, vol. 4, 1994.

[47] A. Newell and H. Simon, *Human problem solving*. Englewood Cliffs, N.J: Prentice-Hall, 1972.

[48] C. L. Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," *Artificiall Intelligence*, vol. 19, 1982.

[49] A. Gupta, C. Forgy, A. Newell, and R. Weding, "Parallel algorithms and architectures for the rule-based systems," *International Conference on Computer Architecture*, pp. 28-37, 1986.

[50] B. Coppin, *Artificial Intelligence Illuminated*, Second Edition ed, 2004.

[51] P. Koton, *Using experience in learning and problem solving*. Cambridge, Massachusetts: Massachusetts Institute of Technology, 1988.

[52] J. J. Daniels and E. L. Rissland, "A case-based approach to intelligent information retrieval," presented at 18th annual international ACM SIGIR conference on Research and development in information retrieval, Seattle, WA, USA, 1995.

[53] P. N. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," presented at 4th ACM-SIAM Sympos. Discrete Algorithms, 1993.

[54] I. Watson, *Applying Case-Based Reasoning: Techniques for Enterprise Systems.*: Morgan Kaufmann Publishers, 1997.

[55] K. B. Taylor, B. G. Nickerson, M. Frize, F. G. Solven, and V. Dunfield, "Use of case-based reasoning to assess patient management in an intensive care unit," presented at Joint Conference COMP and CMBES, Ottawa, 1993.

[56] M. Kaiser and J. Kreuziger, "Integration of symbolic and connectionist learning to ease robot programming and control," presented at ECAI'94 Workshop on Combining Symbolic and Connectionist Processing, Amsterdam, Netherlands, 1994.

[57] M. Sordo, Introduction to Neural Networks in healthcare, http://www.openclinical.org/docs/int/neuralnetworks011.pdf, 2004, July 3

[58] H. Malmgren, M. Borga, and L. Niklasson, "Artificial Neural Networks in Medicine and Biology," presented at ANNIMAB-1 Conference, Göteborg, Sweden, 2000.

[59] R. Salomon and J. van Hemmen, "Accelerating backpropagation through dynamic self-adaptation," *Journal of Neural Networks*, vol. 9, pp. 589-601, 1996.

[60] J. Branke, "Evolutionary algorithms for neural network design and training," University of Karlsruhe, Institute AIFB 322, 1995.

[61] F. V. Jensen, *An introduction to Bayesian Networks*. New-York: Springer-Verlag New York, Inc., 1997.

[62] F. Cozman, "Robustness analysis of bayesian networks with global neighborhoods," Carnegie Mellon University CMU-RI-TR96-42, 1996.

[63] E. L. Eisenstein and F. Alemi, "An evaluation of factors influencing Bayesian learning systems," *Journal of the American Medical Informatics Association*, vol. 1, pp. 272-284, 1994.

[64] D. J. C. MacKay, "Probable networks and plausible predictions - a review of practical bayesian methods for supervised neural networks," *Network: Computation in Neural Systems*, vol. 6, pp. 469-505, 1995.

[65] J. Grimson, W. Grimson, and W. Hasselbring, "The SI challenge in health care.," *Communications of the ACM*, vol. 43, pp. 48-55, 2000.

[66] D. N. Mannai and K. Bugrara, "Enhancing inter-operability and data sharing in medical information systems," presented at International Conference on Management of Data, 1993.

[67] A. Onabajo, I. Bilykh, and J. Jahnke, "Wrapping Legacy Medical Systems for Integrated Health Network," presented at netObjecDays 2003, Erfurt, Germany, 2003.

[68] M. P. Papazoglou and D. Georgakopoulos, "Service-oriented computing: Introduction.," *Communications of the ACM*, vol. 46, 2003.

[69] P. A. Bernstein, "Middlware: a model for distributed system services.," *Communications of the ACM*, vol. 39, pp. 86-98, 1996.

[70] D. Box, *Essential COM. Reading*: Addison-Wesley, 1997.

[71] R. Monson-Haefel, *Enterprise Java Beans*, 2 ed: O'Reily, 2000.

[72] A. Wollrath, R. Riggs, and J. Waldo, "A distributed object model for the Java system," *USENIX Computing Systems*, vol. 9, 1996.

[73] M. C. Little and S. K. Shrivastava, "Implementing high availability CORBA applications with Java," presented at IEEE Workshop on Internet Applications, San Jose, CA., 1999.

[74] Object Management Group (OMG), "The Common Object Request Broker: Architecture and Specification, version 2.4," 2000.

[75] M. Henning and S. Vinoski, *Advance CORBA Programming with C++*. Reading, MA, USA: Addison-Wesley Longman, 1999.

[76] D. Caromel and J. Vayssiere, "A Java Framework for Seamless Sequential, Multi-threaded, and Distributed Programming," presented at ACM Workshop on Java for HighPerformance Network Computing, University, Palo Alto, CA, USA, 1998.

[77] J. M. Shacklette and A. Illian, "CORBA Program Development, Part 2," *Linux Journal*, 1999.

[78] Object Management Group (OMG), "CORBAServices: Common Object Service Specification," Document formal/98-12-31, 1998.

[79] R. Bastide, P. Planque, O. Sy, and D. Navarre, "Formal specification of CORBA services: experience and lessons learned," presented at 15th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, 2000.

[80] Object Management Group (OMG), "CORBA Naming Service," Document formal/97-12-10, 1997.

[81] Object Management Group (OMG), Catalog of OMG CORBAservices Specifications, http://www.omg.org/technology/documents/corbaservices_spec_catalog.htm, 2004, September 15

[82] J. M. Shacklette and A. Illian, "CORBA Program Development, Part 1," *Linux Journal*, 1999.

[83] Object Management Group (OMG), "Trading Object Service Specification," Document formal/2000-06-27, 2000.

[84] P. Jain and M. Kircher, "Lookup Pattern," presented at European Pattern Language of Programs conference, Kloster Irsee, Germany, 2000.

[85] O. M. G. (OMG), "Corba transaction service v1.1.," Document formal/00-06-28, 2000.

[86] Object Management Group (OMG), "Corba security service v1.8.," Document formal/2002-03-11, 2000.

[87] World Wide Web Consortium, http://w3c.org/, 2004, July 19

[88] S. Tsur, S. Abiteboul, R. Agrawal, U. Dayal, J. Klein, and G. Weikum, "Are web services the next revolution in e-commerce?" presented at VLDB Conference, Rome, Italy, 2001.

[89] M. Stal, "Web services:beyond component-based computing.," *Communications of the ACM*, vol. 45, pp. 71-76, 2002.

[90] H. Kreger, "Fulfilling the Web service promise.," *Communications of the ACM*, vol. 46, pp. 29-ff, 2002.

[91] T. Berners-Lee, "Hypertext transfer protocol-- HTTP/1.0.," RFC 1945, 1996.

[92] M. Keidl and A. Kemper, "Towards context-aware adaptable web services.," presented at 13th international World Wide Web conference on Alternate track papers & posters., New York, NY, USA, 2004.

[93] T. Bray and M. Sperberg-McQueen, "Xml language," WorldWideWeb Consortium, 1997.

[94] World Wide Web Consortium, XML Schema Part 1: Structures, http://www.w3.org/TR/xmlschema-1, 2004, September 1

[95] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, and H. F. Nielsen, Simple Object Access Protocol (SOAP) 1.1., http://www.w3.org/TR/2000/NOTE-SOAP-20000508/, 2004, July 21

[96] R. M. Lerner, "At the Forge: Introducing SOAP," *Linux Journal*, 2001.

[97] World Wide Web Consortium, XML Signature Syntax and Processing, http://www.w3.org/TR/xmldsig-core/, 2004, August 1

[98] World Wide Web Consortium, XML Encryption Syntax and Processing, http://www.w3.org/TR/xmlenc-core/, 2004.

[99] OASIS Standards Consortium, Security Assertion Markup Language (SAML) 1.0 Specification, http://www.oasisopen.org/committees/security/, 2004, July 2

[100] P. Hallam-Baker and E. Maler, Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML), http://www.oasis-open.org/committees/security/docs, 2004, October 1

[101] E. Christensten, F. Curbera, G. Meredith, and S. Weeravarana, Web Service Description Language (WSDL), http://www.w3.org/TR/wsdl, 2004, July 21

[102] C. Allen, "WIDL: Application Integration with XML.," *World Wide Web Journal*, vol. 2, 1997.

[103] F. Curbera, S. Weerawarana, and M. J. Duftler, Network accessible service specification language: An XML language for describing network accessible services., http://www.cs.mu.oz.au/~eas/subjects/654/nassl.pdf, 2004, September 12

[104] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the Web services Web: An introduction to SOAP, WSDL, and UDDI.," *IEEE Internet Computing*, vol. 6, 2002.

[105] K. F. Chavda, "Anatomy of a Web service.," *Journal of Computing Sciences in Colleges*, vol. 19, pp. 124-134, 2004.

[106] OASIS Standards Consortium, Universal description discovery and integration (UDDI), http://www.uddi.org/specification.html, 2004, August 16

[107] The Stencil Group, The Evolution of UDDI: UDDI.org White Paper, http://www.uddi.org/pubs/the_evolution_of_uddi_20020719.pdf, 2002.

[108] W. v. d. Aalst and A. t. Hofstede, Workflow Patterns, Standard Evaluation, www.workflowpatterns.com, 2004.

[109] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, I. Trickovic, and S. Weerawarana., Business Process Execution Language for Web Services Version 1.1, http://www-128.ibm.com/developerworks/library/ws-bpel/, 2004, July 12

[110] F. Curbera, R. Khalaf, N. Mukhi, S. Tai, and S. Weerawarana, "The next step in Web services," *Communications of the ACM*, vol. 46, pp. 29-34, 2003.

[111] C. Pahl and C. Casey, "Ontology support for web service processes.," presented at 9th European software engineering conference held jointly with 10th ACM SIGSOFT international symposium on Foundations of software engineering, 2003.

[112] A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, and S. A. McIlraith, "DAML-S: semantic markup for web services," presented at International Semantic Web Workshop, 2001.

[113] M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, and D. L. McGuinness, OWL web ontology language 1.0 reference, http://www.w3.org/TR/owl-ref/, 2004, September 1

[114] O. Corcho and A. Gomez-Perez, "Evaluating Knowledge Representation and Reasoning Capabilities of Ontology Specification Languages," presented at 14th European Conference on Artificial Intelligence, Berlin, Germany, 2000.

[115] S. W. Tu and M. A. Musen, "Representation Formalisms and Computational Methods for Modeling Guideline-Based Patient Care," presented at First European Workshop on Computer-based Support for Clinical Guidelines and Protocols, Leipzig, Germany, 2000.

[116] G. F. S. Foundation, GNU Free Documentation License, http://www.gnu.org/copyleft/fdl.html, 2004, October 6

[117] A. V. Aho and J. D. Ullman, *Principles of Compiler Design*: Addison-Wesley, 1977.

[118] J. Stoy, *Denotational Semantics: The Scott-Strachey Approach To Programming Language Theory.*: MIT Press, 1997.

[119] P. Jones, *The Implementation of Functional Programming Languages*: Prentice Hall, 1987.

[120] G. D. Plotkin, "A structural approach to operational semantics.," Department of Computer Science, Aarhus University DAIMI FN-19, 1981.

[121] N. Chomsky, "Three models for the description of language," *IRE Transactions on Information Theory*, vol. 2, pp. 113-124, 1956.

[122] J. W. Backus, "The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM conference", presented at International Conference on Information Processing, Paris, France, 1960.

[123] E. H. Shortliffe, S. G. Axline, B. G. Buchanan, T. C. Merigan, and S. N. Cohen, "An Artificial Intelligence Program to Advise Physicians Regarding Antimicrobial Therapy," *Computers and Biomedical Research*, pp. 544-560, 1973.

[124] C. Broverman and M. G. Kahn, "Terminology and Information Model Requirements for a Shared Guideline Model," presented at Towards Representations for Sharable Guidelines, Boston, Massachusetts, 2000.

[125] S. W. Tu, J. Campbell, and M. A. Musen, "The structure of guideline recommendations: a synthesis," presented at AMIA Annual Symposium, 2003.

[126] M. Peleg, S. W. Tu, and J. Bury, "Comparing computer-interpretable guideline models: A casestudy," *Journal of American Medical Informatics Association*, vol. 10, pp. 52-68, 2003.

[127] GNU Free Software Foundation, GNU Free Documentation License, http://www.gnu.org/copyleft/fdl.html, 2004, October 6

[128] W. Emmerich, "Software engineering and middleware: a roadmap," presented at International Conference on Software Engineering, Limerick, Ireland, 2000.

[129] R. A. Jenders, G. Hripcsak, R. V. Sideli, W. DuMouchel, H. Zhang, J. J. Cimino, S. B. Johnson, and C. P. Sherman EH, "Medical decision support: experience with implementing the Arden Syntax at the Columbia-Presbyterian Medical Center," presented at Annual Symposium on Computer Applications in Medicine Care, 1995.

[130] D. Chamberlin, J. Clark, D. Florescu, J. Robie, J. Simon, and M. Stefanescu, XQuery 1.0: An XML Query Language, http://www.w3.org/TR/xquery/, 2004, July 15

[131] M. Peleg, O. Ogunyemi, S. Tu, A. A. Boxwala, Q. Zeng, R. A. Greenes, and E. H. Shortliffe, "Using features of Arden Syntax with object-oriented medical data models for guideline modeling.," presented at AMIA Symposium, 2001.

[132] J. Bosak, T. Bray, D. Connolly, E. Maler, G. Nicol, C. M. Sperberg-McQueen, L. Wood, and J. Clark, W3C XML Specification DTD, http://www.w3.org/XML/1998/06/xmlspec-report19980910.htm, 2004, July 1

[133] J. Clark and M. Murata, RELAX NG, www.relaxng.org, 2004, July 26

[134] D. Carlson, *Modeling XML Applications with UML: Practical e-Business Applications*, First ed: Addison Wesley Professional, 2001.

[135] T. Jezler, "IIOP Firewalls," in *Computer Science*, vol. Master in Computer Science. Zurich: University of Zurich, 2001, pp. 100.

[136] R. Bastide, O. Sy, D. Navarre, and P. Palanque, "A Formal Specification of the CORBA Event Service," presented at 4th International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS 2000), California, USA, 2000.

[137] Services Interoperability Organization, http://www.ws-i.org/AboutUS.aspx, 2004, September 15

[138] G. Orth, "The Web Services Framework: A Survey of WSDL, SOAP, and UDDI." Vienna, Austria: Information Systems Institute, 2002.

[139] F. Cabrera, G. Copeland, B. Cox, T. Freund, J. Klein, and T. Storey, Web Services Transaction, http://www'ibm'cm/develperwrks/library/ws-transpec/, 2004, September 23

[140] F. Cabrera, G. Copeland, T. Freund, J. Klein, D. Langworthy, and D. Orchard, Web Services Coordination, http://www.ibm.com/developerworks/library/ws-coor/, 2004, October 1

[141] F. Curbera and R. Khalaf, "The next step in Web services," *Communications of the ACM*, vol. 46, pp. 29 - 34, 2003.

[142] B. L. Donnell, "Object Rule Integration in CLIPS," *Expert Systems*, vol. 11, 1994.

[143] B. Cohen, "Justification of formal methods for system specification," *Software Engineering Journal*, vol. 4, pp. 26-35, 1989.

[144] A. A. F. v. d. Maas, A. H. M. T. Hofstede, and A. J. T. Hoopen, "Requirements for Medical Modeling Languages," *Journal of American Medical Informatics Association*, vol. 10, pp. 146–162, 2001.

[145] G. Schadow, D. C. Russler, C. N. Mead, and C. J. McDonald, "Integrating medical information and knowledge in the HL7 RIM," presented at AMIA Symposium, 2000.

[146] D. Smith, "What Clinical Information Do Doctors Need," *British Medical Journal*, 1996.

[147] H. Healthfield and G. Louw, "New Challenges for Clinical Informatics:Knowledge Management Tools," *Health Informatics Journal*, vol. 5, pp. 67-73, 1999.

[148] A. Moore, "Information rich, knowledge poor," in *KMWorld Magazine*, 2002.

[149] C. Safran, D. M. Rind, and R. B. Davis, "Guidelines for management of HIV infection with computer-based patient's record," *Lancet*, vol. 346, pp. 341-6, 1995.

[150] R. D. Shankar, S. W. Tu, and M. A. Musen, "A knowledge-acquisition wizard to encode guidelines," presented at AMIA Annual Symposium, 2003.