

**HealthWARNer: AN ADVANCE CLINICAL DECISION
SUPPORT SYSTEM, WITH KNOWLEDGE DISCOVERY,
INTEGRATION AND MANAGEMENT FEATURES**

Farhan Gul

(BSc. GIK)

**A THESIS SUBMITTED FOR
DEGREE OF MASTER OF SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
NATIONAL UNIVERSITY OF SINGAPORE**

2005

Summary.....	4
List of Abbreviation.....	6
Index of Figures.....	7
Index of Tables	7
 Chapter 1: Introduction	 8
1.1 Motivation.....	8
1.2 Scope of Research and Contributions	11
1.3 Organization of this thesis.....	12
 Chapter 2: Background.....	 16
2.1 Competing Technologies for base work of HealthWARNer.....	17
2.1.1 Criteria for selecting base work for HealthWARNer	17
2.1.2 Discussion on existing research projects	19
(1) ASBRU.....	19
(2) GEM	21
(3) EON.....	23
(4) GUIDE	25
(5) PROforma.....	26
(6) Protégé.....	28
(7) GLIF	29
(8) Arden Syntax	31
2.1.3 Comparison amongst competing technologies	33
2.1.4 Decision	36
2.2 Research Problem	38
2.2.1 Arden Syntax	38
Implementation	38
Applications of Arden Syntax.....	39
Structure of MLM	39
Versions of Arden Syntax	46
2.2.2 Problems Identified and their Importance.....	47
(1) Poor Healthcare Enterprise Knowledge Integration.....	47
(2) No Process to discover new Knowledge and measure its Efficiency	48
(3) Not easy to Adopt.....	50
(4) Lack of Interest of users in Knowledge Creation Activity.....	50
(5) Outdated MLM file format.....	52
(6) No Separation between Knowledge and Clinical Data	53
(7) Lack of Capability to Perform Disease Surveillance	53
2.2.3 Problem Statement	54
2.3 Comparison of HealthWARNer with the earlier research work.....	55
2.3.1 Knowledge Efficiency Measurement and Discovery	55
2.3.2 Knowledge Integration.....	56
2.3.3 Healthcare Participation.....	56
2.3.4 Sharing Centralized Knowledge Base and Processing Engine	57
2.3.5 Clinical Error Prevention	57
 Chapter 3 Design overview	 58
3.1 Knowledge Efficiency	58

3.2 Knowledge Discovery.....	59
3.3 Knowledge Integration	62
3.3.1 Use of Clinical Standards.....	62
3.3.2 Use of XML Representation	63
3.3.3 Conversion Tool.....	63
3.3.4 Central Knowledge Base and MLM Processing Engine Sharing	63
Chapter 4: Knowledge Creation, Evaluation and Improvement	67
4.1 Addition of <i>Intention, Outcome and Event</i> to evaluate accuracy of knowledge ..	67
4.2 Measuring Efficiency of Knowledge.....	70
4.3 Cycle of Knowledge Creation	72
4.3.1 The role of humans	73
Knowledge Expert	73
Knowledge Translator.....	73
Knowledge Authenticator	74
4.3.2 Computer-Human Interaction (Question and Answers)	74
i. Intention related	75
ii. Outcome related.....	75
iii. Logic related.....	75
4.3.3 Active Knowledge	76
4.3.4 Crude Knowledge	76
4.3.5 Efficiency Measurement	76
4.3.6 MLM History	77
4.3.7 Knowledge Authentication	78
4.3.8 Example Scenario	78
4.4 Improving Healthcare Provider Participation in Knowledge Creation	80
4.4.1 Cycle of Knowledge Creation.....	81
4.4.2 Audit of Treatment Process	81
Chapter 5: Healthcare Enterprise Knowledge Integration	83
5.1 Use of Clinical Standards	83
5.2 Arden Syntax XML Representation.....	85
5.2.1 Use of XML	85
5.2.2 Conversion Tool.....	86
Chapter 6: Centralized Knowledge Base and MLM Processing Engine Sharing	87
6.1 Architecture.....	87
High Level Design	88
Deployment Strategies.....	93
6.2 Centralized Knowledge Base and Process Sharing.....	95
Rationale and Benefits of Using WebServices	95
Design Problems	95
Chapter 7: Implementation.....	99
7.1 Knowledge Acquisition Wizard	99
7.2 MLM Knowledge Management Application.....	101
7.2.1 Roles	101
7.2.2 Actions	101
7.2.3 Create New MLM	102

7.2.4 Modify MLM	103
7.2.5 Import MLM	104
7.2.6 MLM History	105
7.2.7 Translations	106
7.2.8 Users	107
7.2.9 Roles	108
7.2.10 Setting and Logout	108
7.3 Tools	109
Chapter 8: Conclusion.....	110
8.1 Evaluation.....	110
8.1.1 Structure of the System	110
Functionality Completeness.....	110
System Completeness	111
8.1.2 Performance	111
Bottlenecks of HealthWARNer	111
Clinical Testing.....	113
8.2 Test Scenario	113
8.2.1 The Knowledge	114
8.2.2 Knowledge Integration.....	114
8.2.3 Alert	115
8.2.4 Efficiency Measurement	115
8.2.5 New Knowledge Discovery	115
8.3 Contributions.....	117
8.3.1 Knowledge Discovery Process	117
8.3.2 Knowledge Efficiency Calculation	117
8.3.3 Enterprise Knowledge Integration	117
8.3.4 Easy to Adopt.....	118
8.3.5 XML based MLM	118
8.3.6 Improved HealthCare Provider Participation.....	118
8.3.7 Disease Surveillance	118
8.4 Future Research.....	119
8.5 Conclusion	120
REFERENCES.....	123
Appendix A: MLM Examples.....	132
Example 1 ASCII Based text MLM.....	132
Example 2 Anthrax.mlm: XML Based MLM.....	134

Summary

HealthWARNer is an advanced clinical decision support system for minimizing clinical errors of clinicians through clinical alerts generated based on Medical knowledge stored in its knowledge base. It allows clinical knowledge to be shared and reused across healthcare institutions without any manual modifications. HealthWARNer includes processes known as ‘Cycle of Knowledge Creation’ to help discover new knowledge and constantly evaluate existing knowledge. The system automates the process of generating statistical information to rate medical knowledge so that the right judgment can be made in choosing a better treatment process amongst alternatives or to replace a poorly performing clinical procedure with a better one. These statistics are generated each time a patient undergoes a treatment.

HealthWARNer is built upon Arden Syntax, which defines the structure of MLMs. These MLMs hold clinical knowledge for making clinical decisions. MLMs are computer interpretable and have been proven by various studies (discussed later) to reduce chances of clinical errors significantly. Arden Syntax was chosen because it is better suited for generating clinical alerts to prevent clinical errors than other more elaborated methods such as GLIF and PROforma. The latter are typically designed for complex treatment plans of chronics diseases. Arden Syntax is also a mature technology with a simple and efficient knowledge model that has received widespread acceptance from standardization bodies and commercial vendors. We have built upon Arden Syntax in HealthWARNer to make the knowledge format more portable by leveraging on medical standards and the use of more powerful and open IT standards such as WebServices and XML. This allows multiple institutions to share a Central Knowledge Base, which accelerates knowledge discovery as knowledge can be applied, tested and evaluated much more frequently and in a wider scope than in the case of a single healthcare institution. Moreover, the Centralized Knowledge Base

helps to improve the control and management of mission critical clinical tasks such as in detecting and managing disease outbreaks and biological attacks.

We have conducted some tests on HealthWARNer to evaluate whether it has met our research objectives. We found that our new representation of knowledge using clinical standards in XML format can be used to trigger alerts and notify Healthcare providers of possible clinical errors. We tested the mechanism to measure knowledge efficiency and the process to capture new knowledge. The knowledge efficiency results were properly recorded in the history each time the knowledge was executed. We also discovered some new knowledge in the test scenarios, which clearly indicate the success of our concept and HealthWARNer infrastructure for the process of knowledge creation. To test the disease surveillance capabilities of HealthWARNer, we deployed knowledge in a Central Knowledge Base to detect Anthrax exposure in a patient. This knowledge base was shared amongst multiple simulated healthcare institutions. The outcome of this test scenario was successful as the expected warning was generated as soon as we entered dummy patient symptoms similar to Anthrax exposure in either one of the healthcare institutions.

List of Abbreviation

ASTM	American Society for Testing and Materials
CPG	Clinical Practice Guidelines
CPMC	Columbia/Presbyterian Medical Center
DTD	Document Type Definition
GEL	Guideline Expression Language
GEM	Guideline Element Model
GLIF	Guideline Interchange Format
GQAQ	Guideline Quality Assessment Questionnaire
GUI	Graphical User Interface
ICD	International Statistical Classification of Diseases
J2EE	Java 2 Enterprise Edition
LOINC	Logical Observation Identifiers Names and Codes
MLM	Medical Logic Modules
NDC	National Drug Code
PSM	Phase-shifting mask
UDDI	Universal Description, Discovery and Integration
XML	Extensible Markup Language
XSL	Extensible style-sheet Language

Index of Figures

Figure 1.1: Clinical Errors	8
Figure 3.1: Cycle of Knowledge Creation	60
Figure 3.2: HealthWARNer Deployment Scenario	64
Figure 6.1: HealthWARNer Architecture.....	88
Figure 6.2: MLM Alert Notification Window	93
Figure 6.3: HealthWARNer Distributed Deployment	94
Figure 7.1: Knowledge Acquisition Wizard	100
Figure 7.2: Specify new Logic	100
Figure 7.3: Add new MLM	103
Figure 7.4: Edit MLM	104
Figure 7.5: MLM History	105
Figure 7.6: Translators Task List	107
Figure 7.7: Users	108

Index of Tables

Table 1.1 Problem-Solution Summary	14
Table 2.1 Comparison between Guideline Modeling Techniques	34
Table 2.2 Comparison Results	36
Table 2.3 Survey Results	52
Table 4.1 Accumulated knowledge ranking	71
Table 7.1 Roles and Allowed Actions	102

Chapter 1: Introduction

1.1 Motivation

HealthWARNer is intended to reduce unnecessary injury and sometimes the loss of human life due to clinical errors by healthcare professionals. The alarmingly high number of deaths caused by clinical errors prompted this work. The severity of this problem is reflected in the following abstracts of key findings of several recent studies:

- A survey done by the Philadelphia Inquirer and published in September 1999 showed the severity of this problem (Figure 1.1). It reports approximately 120,000 deaths and one million injuries in US in a year due to clinical errors.

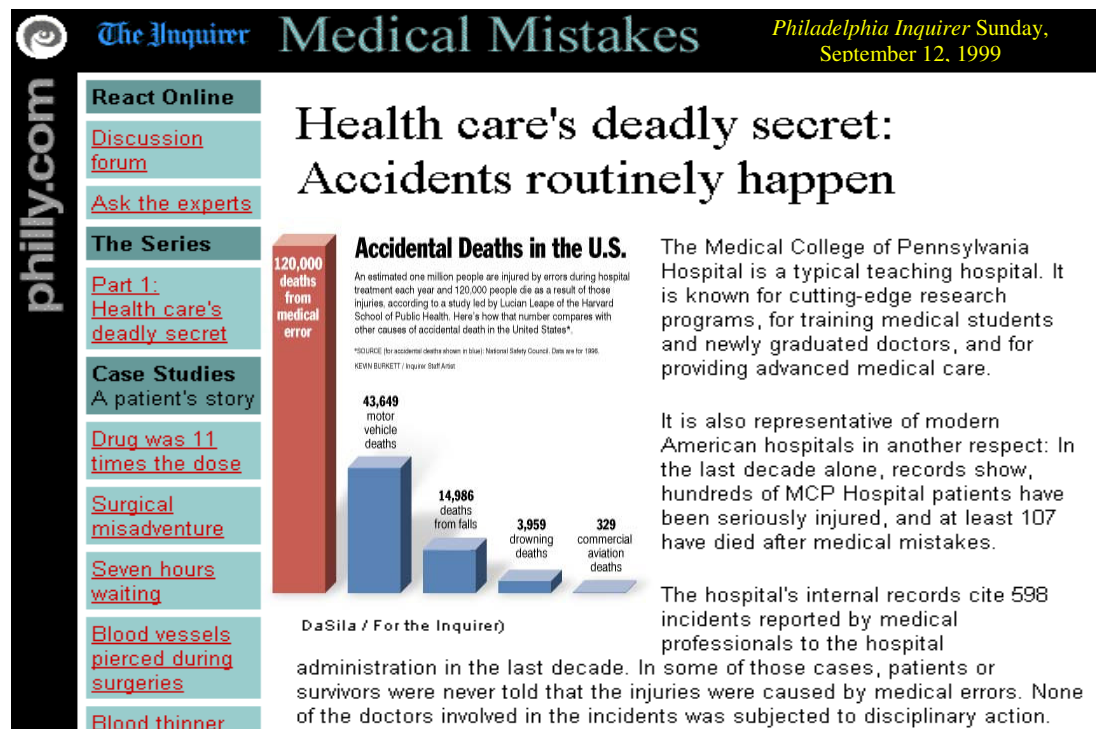


Figure 1.1: Clinical Errors

- An article published in JAMA [27] shows a total figure of 225,000 deaths in a year caused by iatrogenic causes, including unnecessary surgery, infections and adverse effects of medicine.
- Another publication [39] reports a total iatrogenic death figure of 783,936.
- CNN has quoted a report from National Academy of Sciences' Institute of Medicine a figure of yearly deaths between 44,000 to 98,000 in 1999. Note that even 44,000 is a bigger number as compared to annual deaths caused by road accidents, breast cancer or AIDS.

It is worth noting that the figures reported in these reports are very high and the studies were conducted in United States, which has one of the highest expenditure on healthcare with a well-known higher quality of healthcare standards and regulation compared to many other countries. Though there are no similar studies done for Third World countries, we can easily be convinced that the fatal rate due to clinical errors will be higher in these developing countries.

Over time, different means have been developed to moderate the escalating number of deaths due to clinical errors. One of the earliest solutions was the creation of clinical guidelines. A Clinical guideline is defined as a written statement how a certain task has to be fulfilled in a clinical context. But as they were only available in text format that were not interpretable by computer programs, it was awkward for healthcare providers to refer to those documents while they were treating the patients. This inaccessibility of knowledge at the point of care resulted in the ineffectiveness of the guidelines in preventing clinical errors. A major leap was made by the use of Alert bases clinical decision support systems, which has brought clinical rules and guidelines to the point of care. These systems would detect clinical conditions

specified in the knowledge and generate alerts/reminders to healthcare providers of possible clinical errors and provide its recommendations. This minimizes clinical errors by sending alerts and reminders at the appropriate time when it was needed.

Below is a list of studies conducted to evaluate the effectiveness of clinical decision support systems in preventing clinical error and its impact on the cost of treatment

- The study for Perioperative Antibiotic Administration [28] conducted during the period of 1988 to 1994 concluded a decline in perioperative wound infections from 1.8% to 0.9%. It also noted a decline in average number of doses of antibiotics administered from 19 to 5.3, which correspondingly caused a decline in cost per treated patient from \$123 to \$52
- The study on POE with decision support implementation [29] over a period of four years showed that missed-dose medication error rate had fallen by 81% while potentially injurious errors fell by 86%

Based on the above observations and some other publications [61, 62, 63], we conclude that clinical decision support systems can reduce chances of human error and lower medical cost. Here we should emphasize that these findings are for alert based systems implementing simpler guidelines and clinical rules. The same conclusion may not be applied to the use of complex guidelines, which can be executed in several parallel or concurrent plans in various orders.

In reality, it is not sufficient for a solution to prevent clinical errors by expressing knowledge in computer interpretable format and having a clinical decision support system to generate alerts and reminders based on that knowledge. Generating alerts and reminders would help in reducing chances of clinical errors but it can only be as accurate as the knowledge itself,

which is used to generate those alerts and reminders. As clinical practices/procedures are being revised and updated regularly, clinical knowledge stored in such system should also be updated accordingly or it would become obsolete. Therefore a bigger challenge for clinical decision support system would be to find a way through which clinical knowledge can automatically be evaluated and updated in the background based on its effectiveness in treating patients. This is the primary focus of HealthWARNer.

Since such systems already have knowledge in computer interpretable form, it makes a lot of sense if the knowledge representation is standardized so that it can be easily integrated with or reused by other systems. This is similar to the field of Electronics, which is seeing great benefits from creating reusable technologies that can be easily reused and integrated as a component in another system. Unfortunately medical informatics is far behind when it comes to knowledge integration, as it is relatively immature. At the moment, Arden Syntax, the modeling method of clinical guidelines widely accepted by most healthcare standardization bodies and adopted in many commercial implementations (discussed in section 2.1.3) is expressing the knowledge in a format that cannot be used by another healthcare institution without manual changes. This is mainly due to its curly braces problem [36].

1.2 Scope of Research and Contributions

HealthWARNer is designed to generate alerts and reminders to minimize or prevent clinical errors. The approach used is Rule-based methodology for modeling clinical knowledge, which is deemed more effective for handling clinical errors as demonstrated by the related studies mentioned previously in section 1.1. The scope of this research does not include modeling of complex multi-step guidelines, which are more suitable for the treatment of chronic diseases.

The main research objective and our accomplishment are the processes for refinement of clinical knowledge in HealthWARNer. These processes also constantly attempt to discover new knowledge and involve the healthcare providers in the process of discovery. This helps in keeping the knowledge up to date and accurate so that it is more effective for preventing clinical errors. Secondly, we have contributed central processing engine architecture, which can be used by multiple healthcare institutions simultaneously. This extends the scope of traditional Arden Syntax based system across clinical boundaries making it more comprehensive and efficient in detecting clinical errors. Moreover this makes the system easy to adopt, simplifies its management and allows mission critical clinical operations such as the early detections of disease outbreaks and biological attacks.

Additional research objective of HealthWARNer is the extension of Arden Syntax (to be discussed in section 2.2.1). Though Arden Syntax has many useful features and has been fairly successfully applied for moderating clinical errors [28, 29] in several systems, it still has at least two major shortcomings that should be dealt with. First is its curly braces problem [36], which, if is overcome, can lead to better knowledge integration, sharing and reuse. Second, Arden Syntax uses text-based ASCII file format, which is out-dated and inferior for representing knowledge as compared to more advanced format such as XML. We have also achieved these objectives.

1.3 Organization of this thesis

We proposed the design and implementation of HealthWARNer - a prototype of advanced clinical decision support system for minimizing clinical errors of clinicians through clinical alerts generated based on Medical knowledge stored in its knowledge base. Several related research issues and technical problems (listed in Table 1.1) will be addressed in this project as our design objectives. In Table 1.1, the top row of the table lists the issues/problems that we

are addressing in this thesis and the left column outlines our solutions or approaches to resolve them. We are using “√” to link the proposed solution/approach to the respective issue/problem.

Solutions		Problems								
			No process for new Knowledge Discovery	Cannot evaluate knowledge efficiency	No Healthcare Enterprise Knowledge Integration (Share/Reuse knowledge)	Not easy to adopt by healthcare institution & compiler implementation required	Poor healthcare Provider participation in knowledge creation activity	MLM File Format is outdated	No Separation between Data and Knowledge	Cant perform disease surveillance
	Chapter 4	Arden Syntax enhancement (Intention-Outcome, events)	√	√						
		Knowledge efficiency calculation	√	√						
		Audit					√			
		Cycle of Knowledge Creation process	√	√			√			
	Chapter 5	Use of Medical standards			√	√				√
		XML DTD for MLM			√	√		√		
		Conversion Tool for MLM			√	√				
	Chapter 6	Central processing engine (WebServices)	√		√	√			√	√
		Central Knowledge Base			√	√				√
		Security (private / public)			√					
Chapter 7	Knowledge acquisition wizard	√				√				
	MLM Knowledge management Application	√	√		√					

Table 1.1: Problem-Solution Summary

Table 1.1 also outlines the thesis contents, as the solutions/approaches represented in rows are grouped into chapters. Briefly, in Chapter 2, we will have a comprehensive discussion of the clinical decision support systems and their implications on the design of Clinical Practice Guidelines. We will compare various technologies that could be used for the design and discuss how we derived at the conclusion of using Arden Syntax as our base system. After that, we will summarize the shortcomings of Arden Syntax and some useful enhancements that will be carried out in the thesis. At the end of Chapter 2, we will do a comparison of HealthWARNer features with similar work done by earlier researchers.

Chapter 3 provides a design overview of HealthWARNer. We will discuss in details in Chapter 4 to Chapter 6 how HealthWARNer addresses the research issues and problems that have been identified. In Chapter 4, we explain the enhancements made to Arden Syntax in order to improve its efficiency in knowledge representation and for supporting the discovery of new knowledge. Chapter 5 presents the process of Knowledge Integration and explains how HealthWARNer's representation of knowledge is superior to its predecessors. In Chapter 6, we will highlight the benefits of sharing the MLM Processing Engine as WebService and having a Central Knowledge Base, which multiple organizations can share.

In Chapter 7, we will discuss the implementation and evaluation of the HealthWARNer prototype. The knowledge management and acquisition tools provided with HealthWARNer would also be addressed in this chapter.

Lastly, in Chapter 8, we will conclude the thesis. We would also provide some pointers for future research on this topic.

Chapter 2: Background

In this chapter, we present an overview of background work related to HealthWARNer.

Research work in clinical decision systems had started more than two decades ago; groups of researchers have been working on refining a number of technologies for years. There is a variety of available technologies which we could leverage upon, however, the motivations of our design has necessitated some requirements which help us to narrow down the range of technologies that are suitable as the base technology. The requirements eliminated the suitability of most of these techniques leaving us with Arden Syntax.

In the first section of this chapter, we will highlight the scope and requirements of HealthWARNer with which are used to assess the suitability of the various technologies. Then, we will outline in detail the various technologies, examine each of their strengths and weaknesses and explain how we finally short-listed Arden Syntax as the most suitable for our base technology.

Arden Syntax, in addition to being the most suited for our requirements, is also a mature and well-accepted standard technology. However, our research revealed some problems in Arden Syntax, which we think, are important to resolve. In the second section, we will look into these problems and explain why they are important to solve and then draw up a problem statement for HealthWARNer.

Finally, in the last part of this chapter, we would summarize and make a comparison between HealthWARNer and the other similar technologies to demonstrate its superiority in the context of our research objectives.

2.1 Competing Technologies for base work of HealthWARNer

In this section, we will begin with listing out the criteria based on our requirements for HealthWARNer. These criteria will be used to judge which of the earlier work would be more suitable as base work for HealthWARNer. We will also identify and describe related projects, each of which is potentially useful as a base for the development of HealthWARNer. Finally, we will do a comparison based on our criteria to find out which of the competing technologies would be most suitable to be used as foundation for HealthWARNer.

2.1.1 Criteria for selecting base work for HealthWARNer

In this sub-section, we first outline the basic functional requirements of HealthWARNer. These functional requirements are derived from the initial motivations for this project as discussed in section 1.1. In short, our primary motivation is the prevention of clinical errors and in order to achieve this objective, the proven way is to use the rule-based clinical decision support systems, which model simple clinical guidelines. Knowledge Integration, Knowledge Efficiency Measurement and to invent processes for new knowledge discovery are also our motivation from the onset of this project.

The four basic functional requirements of HealthWARNer are as follows:

(a) Clinical Decision Support System

In section 1.1, we discussed that the prevention of clinical errors and lowering its cost are the main motivation factors for HealthWARNer project. We have also highlighted some solutions [28,29] which indicate that alert based [33, 34] clinical decision support system can help

address these issues. It is therefore important for HealthWARNer to have a clinical decision support capability.

(b) Model Clinical Practice Guidelines

All clinical decision support systems have a knowledge component, which they use to make decisions and recommendations. Generally, the knowledge represented in these systems is referred to as CPGs, which are created by healthcare experts inventing best practices for diagnosis and treatment. The CPG can range from simple clinical rules to very complex treatment plans, which are generally used to treat chronic illnesses. As discussed in section 1.1, the simpler guidelines are more suitable for a system, which generates clinical alerts and reminders for error prevention [28, 29, 61, 62, 63]. Therefore HealthWARNer must be able to model simpler guidelines.

(c) Clinical Standard for Knowledge Sharing

Representing Clinical Practice Guidelines in a form that computer can interpret has been addressed in many projects which we shall discuss later in this chapter. The next step for knowledge representation is automated knowledge integration, which would allow CPG knowledge to be shared and reused across healthcare organizations in a seamless manner. To achieve this and to have a widespread use, the knowledge representation has to be accepted as a standard. Hence it is important for HealthWARNer to adopt some standardized methods for clinical knowledge representation.

(d) Commercially Accepted Technology

HealthWARNer should leverage as much as possible on well-accepted methods/techniques that have been proven practical and workable in real world

environment. As discussed in the studies in section 1.1, it is much needed to have decision support systems that can improve healthcare quality and reduce the alarmingly high occurrences of clinical errors. This is an attractive commercial opportunity, which many commercial vendors would like to exploit. We want to base HealthWARNer on a reliable system, so that at the end of the day, HealthWARNer too can be put to some good use, rather than struggle with issues related to the base technologies

2.1.2 Discussion on existing research projects

There are a number of Clinical Practice Guidelines modeling projects that can provide clinical decision support. These are: ASBRU [5, 20, 21], GEM [11, 17], EON [4], GUIDE [44, 45], PROforma [6, 19], *Protégé* [22, 38], GLIF [9, 10, 18], Arden Syntax [16], PRODIGY [7] , GASTON [48, 49], GLARE [50, 51], Prestige [52] and DILEMMA [8]; the following Clinical Practice Guideline modeling methods are still under development: SAGE [53] and DeGel [54]. In the following section, we will provide a summary of some of these technologies, excluding those still under development and the less popular work, of which references are given for further information.

(1) ASBRU

Developed by:

ASBRU was developed under ASGAARD project.

Standard:

Accepted as a standard by ASTM.

Commercial Implementation/vendors:

TBA.

Description and Strengths:

ASBRU [5, 20, 21] is a skeletal plan-representation language to represent time oriented hierarchical clinical guidelines. Using ASBRU, treatment plan can be defined. These plans have specific intentions and can be executed sequentially, in parallel or in any defined order. ASBRU defines mutually exclusive plan instance states like activated, suspended, aborted and completed. Once a plan has been activated, it can only be changed to suspended, aborted and completed state. If a plan is suspended it can be activated again, but this is not possible in the case of aborted or completed. However, a new instance of the plan can be created in this case.

ASBRU plans have five components: preference, intentions, conditions, effects and plan body. It also defines generic guideline plan that are evaluated first to find whether they are suitable to be executed or not. The states defined for them are ignored, considered, possible, rejected and ready. Various conditions are checked before the state is set to ready or rejected.

To handle uncertainty of time duration, starting time and ending time in a treatment plan, ASBRU provides TIME ANNOTATIONS. Using these annotations, minimum and maximum time limits can be specified. This is useful as treatment might show its results sooner or later depending on patient conditions.

A publication [41] concluded that the temporal data abstraction and support for diagnoses and treatment of ASBRU is more superior than its comparable approaches: PROforma, GLIF and EON.

Tools Provided:

AsbruView and AsbrUI

While executing patient data, the duration and success/failure of actions have to be provided to its engine. This interaction is often complex and hard to understand for medical expert so user-interfaces like AsbruView [30, 31] and AsbrUI [32] have been developed to overcome this problem.

Weakness:

Skeletal plan representation is a powerful way of reusing knowledge, however, it makes the interdependencies and composition very complex and difficult to understand [56]

Learning Component:

No such feature available.

Current prominent work in Progress:

- Development of an intermediate representation to visualize the hierarchy of ASBRU language.
- DeGel is extending some work of ASBRU.
- Guideline Markup tool is a tool to convert free text to ASBRU language.

(2) GEM

Developed by:

Yale center for medical informatics.

Standard:

Accepted as a standard by ASTM.

Commercial Implementation/vendors:

None.

Description and Strengths:

GEM [11, 17] uses XML to represent a CPG and is computer interpretable. Using XML as the language gives GEM an advantage of easy XSL transformation to other comparable formats. An example for this is the published work by [42] which attempts to convert GEM encoded guidelines to MLM format. Though the work showed a partial successful transformation, mainly due to differences between GEM and Arden Syntax, there are good chances of successful transformations to other guideline formats with similar characteristics. The (DTD) that defines the structure of GEM representation of a guideline can be seen at <http://ycmi.med.yale.edu/GEM>

Tools Provided:

GEM Cutter

Extracting knowledge and putting that information in GEM format is a tedious process, to overcome this problem, GEM cutter is created. It has a GUI interface to make this process easier.

GEM-Q

GEM-Q [43] is a new tool provided with GEM to evaluate the quality of Clinical Practice Guideline. It is based on GQAQ approach.

Weakness:

One of its weaknesses is that GEM is just an abstraction of the guideline document and therefore needs to depend on extrinsic systems to make it a useful clinical decision support system. It has limited capabilities to tackle ambiguous parts of the guidelines. GEM is not really comprehensive even though it extends the work of several researchers. It may require

extra elements, attributes, and relationships in order to adequately encode guidelines, depending on the guidelines. [11]

Learning Component:

GEM-Q uses Guidelines Quality Assessment Questionnaire (GQAQ) [57]. GQAQ has a guideline quality-rating instrument that comprised of 25 items, which evaluate the development and format of guidelines, identification and summary of evidence, and formulation of recommendations. GEM-Q uses XSL technology to automate this process of quality assessment.

Current prominent work in Progress:

GEM II, which will be more comprehensive and usable.

(3) EON

Developed by:

Stanford medical informatics.

Standard:

None.

Commercial Implementation/vendors:

None.

Description and Strengths:

EON [4] is a set of software components and models for the creation of guideline based application. It also includes guideline modeling and execution system. The guideline model is called Dharma, which includes eligibility criteria, abstraction definition, guideline algorithm, decision models and recommended actions. It also defines goals such as the ideal targeted glucose level. Guideline algorithm can have action steps, decisions, synchronization nodes and can generate recommendations. Protégé-2000 [38] is used for encoding EON guidelines.

EON also models domain ontologies, which is a view of patient data or virtual medical record and other entities like roles in the organization. Patient data is obtained through either database manager or user input.

An advantage of EON is that it allows the reuse of temporal queries and medical domain knowledge.

Tools Provided:

None.

Weakness:

EON does not model execution state in its guideline representation model [58]. Execution states allows a guideline execution thread to suspend, which is a good way to handle long-term guidelines.

Learning Component:

No such feature available.

Current prominent work in Progress:

No information available.

(4) GUIDE

Developed by:

Laboratory for Medical Informatics and University of Pavia, Italy

Standard:

None.

Commercial Implementation/vendors:

None.

Description and Strengths:

GUIDE [44, 45] model is based on Petri Nets and aims to provide integrated knowledge management infrastructure. It uses workflow technology in its multi-level component based architecture to model Clinical Practice Guidelines. GUIDE environment has three main modules, which connect to one another in a loosely coupled fashion using messages. These models are:

1. GIMS - Guideline management system to provide clinical decision support.
2. EPR - Electronic patient record.
3. WFMS / CFMS - Workflow Management System/ CareFlow management system to provide organizational support.

Being based on Petri Net model gives GUIDE an advantage to model complex concurrent processes in sequential, parallel and interactive logic manner. GUIDE provides different view of Knowledge to the various roles defined in the system.

Tools Provided:

To formalize clinical knowledge, GUIDE provides a tool called Guide Editor. Similar to Petri Nets approach, this Editor structures knowledge in the form of a flow chart.

Weakness:

Similar to EON, GUIDE cannot model execution state of the Clinical Practice Guideline.[58]

Learning Component:

One of the objectives of GUIDE was to create new knowledge through constant feedback on guideline acceptance, usability and compliance.

(http://www.labmedinfo.org/research/dsg/decision_support.htm). Details of its implementation are not found in GUIDE publications.

Current prominent work in Progress:

No information available.

(5) PROforma

Developed by:

Advanced Computation Laboratory, Cancer Research UK.

Standard:

None.

Commercial Implementation/vendors:

Arezzo by *InferMed*.

Description and Strengths:

PROforma [6, 19] is a computer interpretable language capable of modeling knowledge in a Clinical Practice Guideline. It has Process description language [40], which uses logic-based approach for decision-making. The PROforma system can maintain and manage clinical procedures and make clinical decisions at the point of care.

PROforma models guidelines as a set of tasks and data items. Task can be a plan, decision, action or enquiry. A 'plan' can further have other tasks including another 'plan'. As the name suggests, 'decisions' are to be made when there are options. 'Action' refers to clinical procedures while 'enquiry' is used to request for further patient data.

Tools Provided:

Arezzo and Tallis.

Weakness:

According to Ruben Meinders in his study about PROforma and ASBRU, (<http://www.ifs.tuwien.ac.at/asgaard/asbru/amsterdam2001/pdfs/voordrppEngl6.pdf>) he concludes that their common drawbacks are:

- Representation of tables
- Simultaneously ending of plans/tasks

- Semantics of preconditions

Learning Component:

No such feature available.

Current prominent work in Progress:

No information available.

(6) Protégé

Developed by:

Stanford Medical Informatics.

Standard:

None.

Commercial Implementation/vendors:

None.

Description and Strengths:

Protégé is an open source ontology development environment. It can be used to develop domain-specific knowledge acquisition system and ontology. It has been used to create and edit content knowledge for knowledge bases. GLIF, PROforma, and PRODIGY use Protégé [22, 38] environment to develop their clinical guidelines. Protégé is also used by the application Dharma to create knowledge for EON. This platform is flexible enough to allow

extension with GUI to work with other knowledge based systems. It can construct domain ontology and provide forms that can be customized for entering domain knowledge.

Tools Provided:

None.

Weakness:

Protégé is meta-tool that can be used to create domain-specific knowledge acquisition systems and model domain ontologies. It has been used to model clinical guidelines in various other techniques. However it lacks the mechanism to execute these guidelines and therefore has to be used along with other systems to create a clinical decision support system. It also performs poorly when attempts are made to link the domain ontology with other modules for collecting and displaying data [55]. Furthermore, it lacks the support to link up ontology concepts with PSM algorithms [55].

Learning Component:

No such feature available.

Current prominent work in Progress:

No information available.

(7) GLIF

Developed by:

Currently under development by InterMed Collaboratory (Stanford Medical Informatics, Harvard, McGill and Columbia University). GLIF had received attention from HL7 and there used to be a special committee there to overlook the development of this standard.

Standard:

None.

Commercial Implementation/vendors:

None.

Description and Strengths:

GuideLine Interchange Format [9, 10, 18] is a formal representation of Clinical Practice Guidelines. GLIF was initially designed for sharing of CPG. Its first published version was GLIF2 and the latest version is GLIF3 (2000). The main difference between GLIF3 and its earlier versions is that GLIF3 is computer interpretable while their earlier versions were not computer interpretable. GLIF defines ontologies for modeling guidelines, medical data and other concepts.

GLIF models guideline like a flow chart, which has steps like clinical decision and action. At each decision step, patient conditions can be checked and branching to some action or another decision step can be made. It supports nesting by allowing sub-guidelines to be added to the guideline flow chart.

GLIF3's major enhancement over earlier version came after they used GEL [36, 37] as expression language. GEL is based on Arden Syntax, which is explained in detail in the next section 2.2.1. As GLIF has an object oriented data model, recently research was done to use GELLO (<http://www.openclinical.org/docs/int/docs/gello.pdf>) as its expression language. As GELLO is also an object oriented expression language, the results were better. GLIF3 also introduces a data layer, which is based on standard medical vocabularies like UML, HL7 Reference Information model. These standards are still under development.

Tools Provided:

None.

Weakness:

It is common to have situations where data items required for guideline execution are missing.

GLIF is unable to handle such situations [60].

Kavanagh [59] discovered some weaknesses of GLIF. These are:

- GLIF coding language is inflexible and requires extensive coding skills to encode guidelines in GLIF format.
- Original integrity of text-based guidelines can be lost during the process of encoding guidelines.

Learning Component:

No such feature available.

Current prominent work in Progress:

The current work on GLIF mainly constitutes the creation of an “Execution Engine” called GLEE [46] and the versioning of guidelines [23]. An execution engine is defined as a software runtime environment, which processes a set of statement and provides it with necessary support functions.

(8) Arden Syntax

Developed by:

HL7 Arden Syntax Special Interest Group and the Clinical Decision Support Technical Committee.

Standard:

HL7, ANSI and ASTM.

Commercial Implementation/vendors:

Implemented by various vendors, namely Micromedex, Siemens, SMS and Eclipsys / Healthvision.

Description and Strengths:

Arden Syntax for Medical Logic Modules (MLM) is a standard for specifying and sharing of medical knowledge [16]. Arden Syntax arose from the need to make medical knowledge available for decision making at the point of care. A system implementing Arden Syntax can generate alert and advice to the healthcare providers to improve quality of healthcare by reducing chances of clinical errors. One of the largest contributions of Arden Syntax is that it standardized the way Knowledge can be integrated into the hospital Information System.

A MLM is a text file holding clinical knowledge according to a specific syntax, called Arden Syntax. A MLM contains a single clinical decision rule, and a typical system implementing Arden Syntax can contain any number of MLMs. The alerts generated by such systems come from these MLMs. Once triggered, MLMs evaluate logical decision criteria and if it holds true, the specified action is performed. Actions usually take the form of sending messages to specified users. MLM is explained in detail in section 2.2.1.

Tools Provided:

MLM library, MLM Syntax checker for windows and many commercial implementations.

Weakness:

Curly braces problem [36] and poor modeling of complex guidelines [58].

Learning Component:

No such feature available.

Current prominent work in Progress:

- Improving of XML schema
- Including Fuzzy logic to enhance Arden Syntax in its version 3
- Improving data type documentation
- Providing better support for imaging
- Providing support for order related to blood products
- Improving messaging

2.1.3 Comparison amongst competing technologies

In this section, we will compare the above-mentioned competing technologies and choose the most suited technology for HealthWARNer based on our requirements. Each of these technologies has their own unique strengths and weaknesses. Many of the technologies names used here like GEM, Arden Syntax and GLIF are sometimes referred as guideline languages in some context. We would like to clarify here that during all comparisons we would be comparing their existing system implementations. We would also like to emphasize here that our eventual choice of technology may not reflect the overall superiority of the technology but rather its superiority in meeting the specified set of criteria for our requirements identified in section 2.1.

(a) Clinical Decision Support System

All of the above-mentioned technologies can be used to create a clinical decision support system. However, some technologies would require additional work or integration with other systems to achieve this objective. But, in principle, they all meet this requirement.

(b) Model Clinical Practice Guidelines

All of the above-mentioned technologies meet this criterion of modeling Clinical Practice Guidelines. Each of them has its unique way of modeling guidelines as summarized in the table below. Dongwen Wang [37] has provides a comparison between most of these technologies (Table 2.1). As this table shows, Arden Syntax is particularly weak when it comes to modeling complex guidelines [3]. But as discussed earlier, we do not really need to model complex guidelines so Arden Syntax capabilities are sufficient for HealthWARNer's needs. Complex guidelines are much needed when modeling "treatment plans" for chronic diseases, while clinical error prevention guidelines are generally simpler in nature. In fact complex guideline modeling methods usually proved too complex and cumbersome [12] to model simpler rules for decision-making and alert generation.

Guideline Model	Representation Primitives				Structure for Primitives		Patient Data Modeling
	Decision	Action	Patient State	Execution State	Temporal Constraints	Nesting	
Arden Syntax [12]	logic slot	action slot	no	no	module invocation	no	no
DILEMMA [13]	state transition	protocol	n/a	procedure state	protocol composition, state transition diagram	protocol	patient record model
EON [14,15]	decision step	action, activity	scenario, activity state	no [§]	flowchart	subguideline	EPR ontology
PROfoma [16]	decision	action, enquiry	n/a	task state	constraints satisfaction graph	plan	n/a
GLIF [17,18]	decision step	action step	patient state step	no	flowchart	subguideline	three-layer domain ontology
Asbru [19]	condition, preference	plan	temporal patterns	plan state	plan-body	plan	n/a
GUIDE [20]	decision	task, wait, monitor	(implicit)	n/a	flowchart	task	relational
PRODIGY [21]	decision	action, activity	scenario	n/a	state transition diagram	subguideline	EPR ontology

Table 2.1: Comparison between Guideline Modeling Techniques

(c) Clinical Standard for Knowledge Sharing

When we compare these clinical guideline modeling technologies based on the level of standardization they have achieved, we find that ASTM has accepted GEM, ASBRU and Arden Syntax, while HL7 and ANSI have only accepted Arden Syntax. Arden Syntax is way ahead of others in terms of its acceptance as a standard. The reason being it is a mature and well-defined technology.

(d) Commercially Accepted Technology

When we go through the list of clinical guideline modeling techniques to look for the ones that have commercial version developed based on these technologies, the results were surprising. Many of the commercial versions available are only based on a handful of these technologies, namely Arden Syntax, ASBRU and *PROforma*. Arden Syntax has been implemented by various vendors including Micromedex, Siemens, SMS and Eclipsys/Healthvision and is installed and running in healthcare institutions like CPMC, LDS Hospital and Intermountain Health Care. Many of the Arden Syntax publications have come from studies done in these hospitals as compared to the other guideline modeling technologies that have never left university laboratory environment. *PROforma* is used by only InferMed (<http://www.infermed.com>) to develop commercial applications like AREZZO. ASBRU has a commercial version by the name of TBA.

The reason for Arden Syntax being accepted commercially is not just because that it is already being accepted as a standard for clinical knowledge representation, but also because its knowledge modeling representation is natural, logical and powerful. Its knowledge representation capabilities will be explained in section 2.2.1. Since Arden Syntax uses a rule-

based approach, the computer program that is written to process this knowledge is much simpler and more efficient as compared to other programs written for other techniques.

A common reason why most of the techniques discussed above do not have a commercial implementation is that they generally model long running complex guidelines with multiple execution threads running simultaneously. This increases the investment and effort in developing their commercial implementation tremendously, hence making them less viable.

2.1.4 Decision

Based on the comparison study, three out of the eleven technologies stand out after applying our four criteria. In terms of suitability for our project, we would rate Arden Syntax as the most suited, as illustrated in table 2.2. This table summarized the results of criteria three and four only, as all the technologies meet the first two criteria.

	Clinical Standard	Commercially Accepted
Arden Syntax	HL7, ANSI and ASTM	4 Vendors
PROforma	None	1 Vendor
GEM	ASTM	None
Asbru	ASTM	1 Vendor
Rest of the technologies	----- None -----	

Table 2.2: Comparison Results

We did not choose *PROforma* mainly because it is not accepted as a standard. Though one commercial vendor has implemented it, the main focus is on specific chronic illnesses, while Arden Syntax caters for a broader range of treatments.

ASBRU is stated to have this commercial version called TBA, but there is no information available or any evidence to show that this system is running in any healthcare institution. None of the commercial vendors has implemented GEM. Both ASBRU and GEM have not been accepted by HL7 or ANSI, which are more recognized in the field of medical informatics as compared to ASTM.

Based on the set criteria and the objectives of HealthWARNer, Arden Syntax proves to be the best choice available for the foundation of our work. Multiple commercial vendors and standardization bodies have adopted and accepted it. Besides meeting all the criteria, Arden Syntax is receiving special attention from HL7. There is a special interest group in HL7, with members comprising of Arden Syntax vendors and healthcare institutions, working for the further development of this standard. Arden Syntax might not be the best choice for modeling complex guidelines, but it has proven itself when it comes to modeling simpler guidelines and decision rules to prevent clinical errors. We have chosen Arden Syntax as base work to have a solid and reliable foundation. According to Samson W. Tu:

“Arden Syntax is not infant technology; it has gone through a decade of evolution and has been continuously refined by multiple implementations by commercial vendors and healthcare institutions. It is a mature standard which has proved itself in improving healthcare by reducing chances of clinical errors and reducing the cost of preventing errors.”

Though we base our research on Arden Syntax, we fully understand and appreciate the significance of the contribution of the other competing technologies for modeling clinical guidelines in general. They each have their own unique strengths but these strengths are not relevant to our research objective.

2.2 Research Problem

As discussed above, we will base HealthWARNer on Arden Syntax. Arden Syntax is a mature and well-accepted standard technology but our research has uncovered some of its limitations that we will need to resolve in order to present a better solution in HealthWARNer. In the first sub-section, we will explain the details of Arden Syntax that are necessary to understand its limitations. Later, we will explain specifically what the problems are and why it is important to solve them. Finally, we will state our problem statement.

2.2.1 Arden Syntax

In this section, we will focus on Arden Syntax implementation and explain how it works. Since we have selected Arden Syntax as the foundation for HealthWARNer and the fact that we have attempted to address some of the deficiencies and weaknesses related to Arden Syntax, it is therefore essential to elaborate further on Arden Syntax.

Implementation

A typical high-level view of Arden Syntax implementation would include two core components. The first component would be the knowledge, which comprises of a set of MLM and the second component would be the run-time engine, which would include the compiler for MLMs and the run-time environment for MLM execution. Arden Syntax is the syntax of MLMs. It describes the structure of knowledge, in other words, how knowledge can be expressed in the form of a MLM. Arden Syntax only standardizes the knowledge component

but not the engine. Each institution that requires the processing of the MLM needs to implement its own compiler and run-time engine.

Applications of Arden Syntax

Some examples of the areas where this MLM knowledge is applied:

- Generating clinical alerts
- Performing Interpretations & diagnoses
- Screening for clinical research
- Performing administrative support
- Performing quality assurance functions

Structure of MLM

In this sub-section, we will describe the structure and syntax of MLM and give short code examples, wherever necessary. Most of the examples have been extracted from CPMC (Columbia-Presbyterian Medical Center) shared library and John Dulcey presentation at AMIA Fall Symposium, 2001. We will go into greater level of details here as some of the research problems discussed in this thesis later relates to this syntax of MLM.

Three Main Slots

MLM is a stream of structured text stored in an ASCII file. The statements present in a MLM are referred to as slots and slots are grouped into three categories/slots, namely:

Maintenance: As the name indicates, it groups maintenance related information about the MLM. The slots present in maintainace are Title, Filename, Version, Institution, Author, Specialist, Date and Validation.

Library: It holds description about the purpose and knowledge of the MLM. It also provides citations and links about the knowledge. The slots present in Library are Purpose, Explanation, Keywords, Citations and Links.

Knowledge: This is the main computer interpretable part of the MLM. It would describe when the MLM is supposed to be fired, what would be its logic to make the decision and the action that would follow. The slots present in Knowledge are Type, Data, Priority, Evoke, Logic, Action and Urgency.

Syntax of each slot

Each slot has a slot name and a slot body, like:

maintenance:

slotname: slot-body;;

slotname: slot-body;;

...

library:

slotname: slot-body;;

...

knowledge:

slotname: slot-body;;

...

end:

Now we will discuss some of the important slots in *Knowledge* category/slot.

Data Slot

A MLM has to access some patient data in order to apply its knowledge and make a decision. Arden Syntax allows MLM to have a very flexible way to query patient data. It allows the use of curly braces [36] to put in any query language used by the institution. This allows the institution to work with any kind of database (SQL, object oriented or XML etc). Arden Syntax does not define any structure within the curly braces. Within a data slot, user can use statements like 'read' to access data from its database. For example

Data:

```
creatinine := read {'dam'="PDQRES2"};  
last_creat := read last {select "OBSRV_VALUE"  
from "LCR" where qualifier in  
("CREATININE","QUERY_OBSRV_ALL")};;
```

Take note that there are two types of read statements being used here, 'read' and 'read last'. Read statement without an operator will fetch a list of values. If a specific value is needed, then Arden Syntax provides a list of operators like: first, last, max, min, count, average, sum, etc.

Evoke slot

Evoke slot defines the triggering condition of a MLM. When all the conditions specified in the evoke slot are met, the MLM would be executed/fired. It allows users a lot of flexibility in the way a MLM can be fired. The conditions would be:

- On the occurrence of an event
- After an interval of time after the occurrence of an event
- Periodically after an event, until a fix duration of time or the occurrence of another event or condition

- When called by another MLM

Some examples:

- evoke: 3 days after time of creatinine_storage;;
- evoke: every 1 day for 7 days starting at time of creatinine_storage;;
- evoke: every 1 day starting at time of K_storage until K>=3;;

Logic slot

Once a MLM evoke condition is met, logic slot is executed. As the name suggests, it holds the decision logic. It will fetch the data and process it according to the logic and then conclude false or true. In case the result is false, MLM does not take any action, otherwise, the specified action will take place.

Arden Syntax has most of the programming language constructs available in other languages like C and Pascal etc. In a logic slot, user can use if-then-else statements to check for various condition and branch through them. It also allows nesting of these statements to model complex medical logic. If-then-else structure is like:

```
if <expression1> then
<One or more lines of Arden Syntax code>
endif;
```

```
if <expression1> then
<One or more lines of Arden Syntax code>
elseif <expression2> then
```

```

<One or more lines of Arden Syntax code>
elseif <expression3> then
<One or more lines of Arden Syntax code>
...
elseif <expressionN> then
<One or more lines of Arden Syntax code>
else
<One or more lines of Arden Syntax code>
endif;

if <expression1> then
<One or more lines of Arden Syntax code>
else
<One or more lines of Arden Syntax code>
endif;

```

Note: <expression> is a condition, which can evaluate to either true or false.

An example of logic slot using If-then-else structure:

```

logic:
if last_creat is not present then
    alert_text := "No recent creatinine available. Consider
                    ordering creatinine before giving IV contrast.";
conclude true;
elseif last_creat > 1.5 then
    alert_text := "This patient has an elevated creatinine."

```

Giving IV contrast may worsen renal function." ;

conclude true;

else

conclude false;

endif;;

Logic slot also allows 'while' and 'for' looping:

while <expression> do

<One or more lines of Arden Syntax code>

enddo;

for <expression> do

<One or more lines of Arden Syntax code>

enddo;

Action slot

As mentioned earlier, action slot is executed when logic slot concludes true. In an action slot user can specify to:

- Write a message to screen
- Store a message in a file
- Call another MLM

Other programming constructs

Time and Duration

‘Time data’ refers to points in time and ‘duration’ refers to an interval of time. Arden Syntax allows numerous comparison operators for them, for example:

<time> is before <time>

<time> is after <time>

<time> is equal <time>

<time> is within <time> to <time>

<time> is within <duration> preceding <time>

<time> is within the past <duration>

Examples:

- time of potassium is before 1996-09-16T00:00:00
- time of potassium is after time of digoxin level
- time of potassium is within 1 hour preceding time of creatinine
- time of potassium is within the past 5 days
- time of potassium < (now - 3 days)

Arden Syntax provides the following operators which are the essential parts of any programming language:

- *assignment operator*
- *conditional operators*
- *comparison operators*
- *logical operators: and or not*
- *mathematical operators*

MLM Examples

Now that we have given an introduction to Arden Syntax for MLM, we will present some simple examples of the type of knowledge that can be specified and placed in a MLM.

Appendix A shows some complete MLM codes, which are used to model such knowledge.

- Send an alert to a healthcare provider to warn him or her when the patient's hematocrit is very low or falling rapidly.
- Clinical evidence affirms that elevated total cholesterol (and especially elevated LDL cholesterol) correlates with an increased risk of coronary artery disease. Furthermore, lowering total (and LDL) cholesterol values reduces CAD risk. The National Cholesterol Education Program advises that a total cholesterol (and HDL cholesterol) level should be checked every 5 years for normal adults over 20 years of age.
- Indomethacin and sulindac may cause or worsen renal insufficiency. In addition, a typical dose of the NSAID may require adjustment when it is administered to a patient who already has renal insufficiency. This module sends an alert if one of this class of drugs is ordered for a patient who has laboratory evidence of renal insufficiency to help ensure that appropriate action (e.g., dosage adjustment) is taken if needed.

Versions of Arden Syntax

The current fully approved version of Arden Syntax is 2.1. Work is in process to finalize version 2.5, which would include:

- An improved version of XML schema
- An improved data type documentation
- Better support for imaging
- Support for order related to blood products

- Better messaging

The Special Interest group at HL7 intends to include Fuzzy logic to enhance Arden Syntax in its version 3.

2.2.2 Problems Identified and their Importance

Though Arden Syntax is a relatively mature technology, there are still some problems and some areas, which have not been addressed by earlier research. In this sub-section, we will discuss in details the problems and explore the areas, which have not been addressed by earlier researchers. We would also elaborate why it is important to address these issues. In the end we would derive our problem statement, which would lay the foundation of HealthWARNer.

(1) Poor Healthcare Enterprise Knowledge Integration

One of the main objectives of Arden Syntax was to create knowledge that can be shared [16] amongst institutions. This objective is still not met because of the induction of curly braces. As described earlier, in Arden Syntax, clinical data is fetched in the data slot, which allows the MLM writers to write any sort of query language within the curly braces. There were two reasons for keeping this. Firstly, it allows the MLMs to work with any type of database with ease, as any sort of query can be placed within the curly braces. This allows the institution to integrate Arden Syntax implementations with object-oriented; XML or SQL based types of EMR databases. The second reason was that at the time Arden Syntax was designed, there were not enough medical standards to express clinical data. Curly braces technique worked well within an institute but if the knowledge had to be transferred to another healthcare institute, manual changes will inevitably be required [15, 36]. Another shortcoming of Arden Syntax is that it does not define events. Arden Syntax is used for creating event driven systems but the lack of event definition, makes it pretty awkward to use. The reason why

events are not defined is that they are closely related to patient data. As data comes from curly braces, which is not structured, events also cannot be structured properly.

Addressing these issues is very important; otherwise, true knowledge sharing cannot be achieved. Ideally, a MLM should be like “plug n play” knowledge, which should be easily integrated into the system. Users should be able to download, install and start using MLM with a few clicks rather than having to go through medical dictionaries and documents to translate the clinical data and the queries to their own representation. Arden Syntax’s current status of knowledge sharing is not only time consuming but also a continuous repetition of the same task.

(2) No Process to discover new Knowledge and measure its Efficiency

With the exception of GEM and GUIDE, all the technologies including Arden Syntax, were designed to express clinical knowledge for sharing and to make it computer interpretable. Arden Syntax, as well as most of the mentioned technologies met most of the intended purposes but if one tries to judge whether their knowledge has proven to be useful or not, all the process has to be carried out almost manually. GEM-Q [43, 57] is a new tool provided with GEM to evaluate the quality of Clinical Practice Guideline. It is based on GQAQ (Guideline Quality Assessment Questionnaire) approach. The problem with this approach is that it is based on a questionnaire rather than the outcome of treatment on the patient, which is a better approach [43]. GUIDE allows new knowledge to be generated through constant feedback on guideline acceptance, usability and compliance. However, details of whether it is accomplished manually or automatically is not published.

Arden Syntax specifications or implementations, like most of the other technologies, do not have any constructs to help improve knowledge automatically or even to judge whether the knowledge is correct. The closest construct that exists in Arden Syntax is the purpose slot, but it is not structured and can only be used in a manual process to judge whether the MLM is meeting its expectations. There might be numerous alerts or decisions made by a MLM for a number of patients. Processing this information manually can be extremely long and complex and not feasible in most cases. Similarly, when it comes to knowledge discovery, there is no process or technique provided. The only possible way to increase the size of knowledge base is to manually encode a MLM and deploy it in the system.

We discovered this issue and believe that addressing it could lead to great benefits. The knowledge in the MLMs is used to point out mistakes in clinical decision and ultimately reduce chances of clinical errors. But what if that knowledge itself is not accurate? The results can be disastrous, so it is absolutely essential to have a mechanism to measure knowledge efficiency all the time. Manual process is too slow to be a solution so it has to have a built-in feature to automatically validate MLM continuously. The evaluation of knowledge efficiency will also play an essential role in new knowledge creation. Knowledge in a clinical decision support system gives it the intelligence to make decisions. The more knowledge a clinical decision support system has, the greater its intelligence would be, resulting in greater benefits. The system relying completely on manual entry of knowledge would definitely grow very slowly as compared to a system which complements manual entries with automated processes to add and improve knowledge.

(3) Not easy to Adopt

Arden Syntax only defines and standardizes the Syntax of MLM; the compiler and run time environment implementation has been left to the choice of healthcare institution. Each institution has to implement its own compiler and the runtime environment needed to run this system, which is not the end of its problems. Each MLM would also require manual modifications before it can be used. This also hurdles the pace of research on this area, as each researcher has to write a compiler first in order to start their research. There have been a few compilers shared by some institutions but they are just executable files specific to that particular institution. This can only be used to check the syntax of MLM written for that institution and nothing more than that, which is almost of no use. Each institution has to put in its finances and efforts to reinvent the wheel.

Having a run-time engine, which can be shared amongst various institutions, can contribute not only to saving finances, time and efforts, but also to accelerating the growth of knowledge in the Knowledge Base. It would make a great difference if this sharing is done in a way where all applications can use it independent of their platform and the language used to write them.

(4) Lack of Interest of users in Knowledge Creation Activity

Initial resistance towards knowledge management system is generally observed in many industries [26]. This led us to find out whether doctors would be interested in contributing to Arden Syntax knowledge. There is no publication addressing this issue, so we went through the shared MLMs of CPMC (Columbia-Presbyterian Medical Center) to conduct a short survey. We found that 94 % of the MLMs are written by a small group of people, who happened to be the most prominent researchers of Arden Syntax. Others made only 6% of the contribution. CPMC is a large medical center and this system has been running there for quite

some time, but other doctors not involved in Arden Syntax research do not seem to be interested in contributing to knowledge. We found a clear pattern indicating their lack of interest in contributing to knowledge.

Survey

Location

We did a survey at the shared CPMC Medical Logic Module Library

(<http://cslxinfmtcs.csmc.edu/hl7/arden>).

Method

We randomly selected 110 MLMs from the total of 250 to 300 MLMs that they had shared.

We took note of the author's name for each of them and categorized them based on Arden Syntax publication authors and others.

Results

Based on our sampling, 94% of the MLMs have been written by prominent researchers who have authored numerous Arden Syntax publications. The details of the results are shown in Table 2.3.

	Name of MLM Author	Number of MLM published
Authors of Arden Syntax publications	Robert A. Jenders	28
	George Hripcsak	19
	Eric H. Sherman	23
	Justin B. Starren	18
	Robert V. Sideli	15

		Total = 103
Have not seen their name in	Cynthia Chen	3
Arden Syntax Publications	Vadim Potievski	4
		Total = 7

Table 2.3: Survey Results

Knowledge is distributed all across an institution [1] and a good Knowledge Management System also needs to explore ways to gather this knowledge efficiently and effectively. There is a need to find a better way to obtain greater participation and involvement of doctors in the knowledge creation activity.

(5) Outdated MLM file format

As mentioned above, each healthcare institution has to implement its own compiler; on top of this, the MLM is an ASCII based text file. It is much more complex to write a compiler for such files as compared to XML file. Besides writing compiler, it is also difficult to search or to translate this document into other formats. Arden Syntax standardization body has already realized the importance of having XML format. They have decided on the setting of four levels or stages for conversion to XML, which are:

Level 1: Include only the whole text based MLM in one XML tag.

Level 2: Tag each slot except but logic slot, which will remain the same.

Level 3: Tag keywords in logic & data slots.

Level 4: All the statement in every slot will be tagged.

Unfortunately, the work has been pretty slow on this much-needed change. So far, only work up to level 2 has been completed. This excludes the most significant section of MLM- the logic slot that enables it to make clinical decision.

(6) No Separation between Knowledge and Clinical Data

As discussed briefly at the end of section 2.3.4, there is a need for a central engine, which multiple institutions can share. In order to do this, the structure of MLM proves slightly rigid. The reason is simply: 'it is not designed to be processed in a shared environment', as needed by HealthWARNer. HealthWARNer requires data slot to be designed in such a way that data can be plugged into and removed as needed.

(7) Lack of Capability to Perform Disease Surveillance

Disease surveillance is defined as an observational study that involves continuous monitoring of disease occurrence within a population. Arden Syntax has generally most of the features required for creating a disease surveillance application, but its scope is limited to a single healthcare institution, which makes it hard to detect a disease outbreak. Using the current Arden Syntax implementation model, even if all the healthcare institutions are equipped with disease surveillance MLMs, the system would still not work smoothly as there would be no communications links between these institutions.

The benefits of having a disease surveillance application can result in early detection of disease outbreak and biological attacks. Though it cannot prevent the unfortunate from happening, an early warning could help control and contain situation in a far more effective way.

2.2.3 Problem Statement

HealthWARNer has to be a rule-based advance clinical decision support system designed primarily to reduce incidences of clinical errors. The advance features should include processes for continuous knowledge evaluation and discovery. In terms of knowledge representation, it should extend Arden Syntax to provide Healthcare Enterprise Knowledge Integration, so that true knowledge sharing can be achieved. Moreover, HealthWARNer should improve on Arden Syntax knowledge format, which is a text-based ASCII file. In terms of infrastructure, it should be able to integrate with any application written in any language and running on any platform, so that it can be easily adopted by any healthcare institution, not requiring them to create Arden Syntax compiler and run time engine. Since a clinical decision support system integrates with clinical applications and EMR, it would be good to have disease surveillance capabilities in HealthWARNer. Lastly, to stand out from the other existing clinical decision support systems, it should also improve healthcare provider participation in the process of knowledge creation.

2.3 Comparison of HealthWARNer with the earlier research work

In this section, we will make a comparison between HealthWARNer and the existing technologies mentioned in section 2.3 to highlight our research contributions. Please note that in order to understand the following sections completely further knowledge about HealthWARNer design is necessary. Please read to chapter 3 and chapter 6 before reading this section. Also note that many of the technologies names used here like GEM, Arden Syntax and GLIF are sometimes referred as guideline languages in some context. We would like to clarify here that during all comparisons we would be comparing their existing system implementations.

2.3.1 Knowledge Efficiency Measurement and Discovery

HealthWARNer has the capability of measuring the efficiency of knowledge and a process to capture new knowledge to be discussed in detail in Chapter 4. Amongst the existing technologies, we have found only two other technologies to have similar capabilities, which are GEM and GUIDE. GEM has a tool called GEM-Q [43, 57] to evaluate the quality of Clinical Practice Guideline. GEM-Q uses Guidelines Quality Assessment Questionnaire (GQAQ) [57]. GQAQ has a guideline quality-rating instrument that comprised of 25 items, which evaluate the development and format of guidelines, identification and summary of evidence, and formulation of recommendations. GEM-Q uses XSL technology to automate this process of quality assessment. However, this approach does not take patient outcome into consideration like our approach, which is superior [43]. One of the objectives of GUIDE was to create new knowledge through constant feedback on guideline acceptance, usability and

compliance. (http://www.labmedinfo.org/research/dsg/decision_support.htm). Details of its implementation are not found in GUIDE publications.

2.3.2 Knowledge Integration

The most important factor for knowledge integration is the acceptance of the technology as a standard by established standardization bodies. Amongst the technologies discussed, ASTM has accepted GEM, ASBRU and Arden Syntax, while HL7 and ANSI have only accepted Arden Syntax. Arden Syntax is the most accepted standard amongst all of the other technologies. Arden Syntax has a major problem called the curly braces problem, which prevents direct re-use of its knowledge. HealthWARNer follows Arden Syntax representation of knowledge with enhancements to remove curly braces problem to make its knowledge more integratable.

2.3.3 Healthcare Participation

This area of research has not been addressed by any of the publications for the technologies mentioned above. We conducted a survey described in section 2.2.2 to detect the level of participation by healthcare providers for Arden Syntax implementation at CPMC. Since most of the other technologies do not have commercial implementations, which share knowledge bases, we could not extend our survey to them. Our conclusion from the survey was that only 6% of knowledge contribution was made by healthcare providers. This reflects the lack of attention paid to encouraging healthcare providers to participate in knowledge creation activities in Arden Syntax. To be discussed in section 4.4, HealthWARNer tries to address this issue using a couple of techniques.

2.3.4 Sharing Centralized Knowledge Base and Processing Engine

HealthWARNer adopts a unique architecture to expose its processing engine and Central Knowledge Base through a WebService. This approach is new to Arden Syntax and the other technologies, and gives HealthWARNer an additional edge for disease surveillance and accelerates knowledge improvement and discovery. Such features are not available in any of the other technologies.

2.3.5 Clinical Error Prevention

The techniques mentioned above use different approaches to model clinical guidelines. Arden Syntax uses Rule-based approach, PROforma uses Logic based approach, PRODIGY uses Network based approach and GUIDE uses workflow (Petri Nets) technology. Each of these approaches has its own strengths and weaknesses. However, when it comes to the prevention of clinical errors, Rule-based approach has been proven to reduce clinical errors [28, 29, 61, 62, 63]. HealthWARNer follows the same rule-based approach as its predecessor- Arden Syntax. The main difference between HealthWARNer and Arden Syntax rule-based approach is in the scope. In HealthWARNer, rules can cross healthcare enterprises' boundaries and can be used to enforce guidelines and policies for a group of participating healthcare enterprises; unlike Arden Syntax, which is only limited to a individual healthcare enterprise.

Chapter 3 Design overview

In this chapter, we will present an overview of the design of HealthWARNer.

We will discuss how it achieves knowledge integration by the use of accepted standards. We will also look at how it uses Cycle of Knowledge Creation to measure knowledge efficiency, improve knowledge further and even discover new knowledge. Furthermore, we will elaborate on the architecture of the Central Processing Engine and Knowledge Base, which will provide us with the capabilities of disease surveillance.

3.1 Knowledge Efficiency

One of the major design objectives of HealthWARNer is to measure the efficiency of knowledge. The approach adopted by HealthWARNer is fully automated and it rates the knowledge within a MLM every time it is applied to the patient. The results are stored in history and can be used either to establish the effectiveness of the knowledge or to compare the treatment specified in the MLM with an alternative treatment.

In order to measure knowledge efficiency, we have incorporated some new constructs in Arden Syntax. These constructs are ‘intention’ and ‘outcome’ of the knowledge specified in the MLM. Since each MLM describes the expected outcome or the desired patient state, we can compare it with the actual patient state observed after a treatment is applied on the patient. We have designed a mechanism, which scans the EMR for the expected outcome and after the comparison with the desired results, rates the knowledge.

The syntax of ‘intention’ and ‘outcome’ will be discussed in details in Chapter 4. In brief, it describes the patient state, which will be achieved, and the duration of time for which the system will wait for the patient condition to come to this state. Alternatively, this time duration can also be replaced with a defined ‘event’, which will indicate the time to check the patient condition. At the occurrence of the ‘event’ or the time duration deadline, the system goes into the EMR and fetches the patient state. The process of accessing the patient state from the EMR is rather complex and will only be explained later in subsequent chapter. The combination of the outcome and the time taken for the achieving this outcome will determine the rating of the knowledge in MLM. Each time a MLM is fired, it is rated and stored in the history according to the following five ranking: ‘Excellent’, ‘Need Improvement’, ‘Poor’, ‘Undetermined’, and ‘Waiting for results’.

3.2 Knowledge Discovery

We have introduced a process in HealthWARNer to help discover new knowledge, which we termed ‘the Cycle of Knowledge Creation’. The main difference between earlier approaches [Arden Syntax references] and ours is that we model knowledge as a dynamic component of the system instead of a static component. This Cycle of Knowledge Creation never assumes knowledge to be final and correct. It is an iterative process, which improves knowledge constantly. This process also attempts to capture new knowledge from the healthcare providers. This Cycle of Knowledge Creation has been built upon Arden Syntax; however, it can be extended to other clinical support systems.

Figure 3.1 illustrates the various stages of this Cycle of Knowledge Creation.

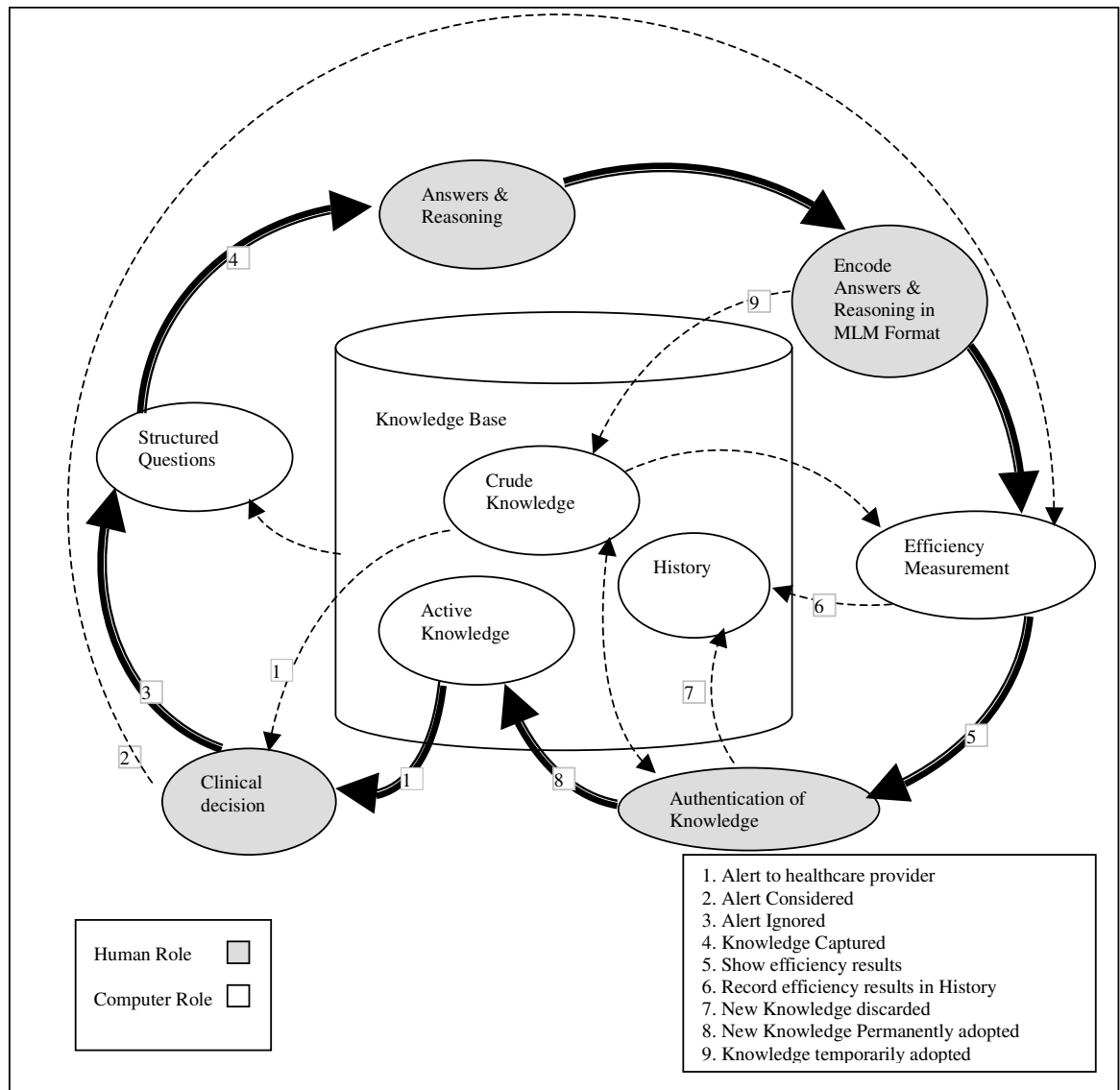


Figure 3.1: Cycle of Knowledge Creation

The knowledge discovery aspect of this cycle is based on the hypotheses that whenever a healthcare provider ignores an alert or a reminder generated from the clinical support system, he/she has some piece of knowledge in his/her mind that is not present in the system. This process attempts to capture that piece of new knowledge from the healthcare provider and evaluate it before it is fully accepted as authentic knowledge in the system.

As shown in step 1 of figure 3.1, alerts are sent to the healthcare provider who would either consider or ignore the alert to make the clinical decision. In case the alert is ignored, the process tries to capture new knowledge from the healthcare provider.

In steps 3 and 4, the healthcare provider is presented with some structured questions so as to capture the new knowledge in certain raw format. The Knowledge Acquisition Wizard presents these questions. This wizard and the details of the questions will be further explained in later chapters. In short, the questions relate to the intention, outcome and the logic of the healthcare provider in making that clinical decision whilst ignoring the alert. HealthWARNer also includes a Knowledge Management Application, which provides a workflow for this Cycle of Knowledge Creation. This Knowledge Management Application adds the knowledge captured from the healthcare provider to the tasks list of the knowledge translator whose role is to translate the knowledge expressed in English to MLM format.

This newly captured knowledge is put in place in the system as Crude Knowledge. Crude Knowledge, as opposed to Active Knowledge, is not authenticated and hence the alerts generated from it indicate its status. The knowledge within both Active and Crude Knowledge repositories are evaluated for its effectiveness each time the knowledge is used to generate an alert or make a recommendation. The new knowledge that has been added into the Crude Knowledge repository will be evaluated when the patient outcome is observed by the system. The results of the evaluation will be stored in the MLM history for later review by the knowledge authenticators. A new piece of knowledge might have to go through this Cycle of Knowledge Creation several times before it has enough rating information in the MLM history to help knowledge authenticator to authenticate the knowledge. Once the knowledge is authenticated, it is moved to the Active Knowledge repository.

3.3 Knowledge Integration

3.3.1 Use of Clinical Standards

As mentioned earlier in Chapter 2, HealthWARNer is built upon Arden Syntax and Arden Syntax has the intrinsic problem of ‘curly braces problem’. This problem is created by the induction of curly braces in the data slot of the MLM. There is no structure defined for the query within these curly braces. This gives the institution the flexibility to use any type of database (relational, object-oriented or XML etc). However, this apparent flexibility prevents the MLM knowledge to be freely integratable into the clinical support system.

To render the knowledge more integratable, we have incorporated some clinical standards to replace the curly braces in HealthWARNer knowledge representation. The reason for choosing more than one standard is that there is no one particular standard which is comprehensive enough to describe all the various types of patient-related data that can be possibly referenced within a MLM. We have used LOINC - <http://www.loinc.org> to represent laboratory observations, ICD and Related Health Problems - <http://www.who.int/whosis/icd10> for diseases classification and NDC - <http://www.fda.gov/cder/ndc> for drug codes. These are codes set and classified by standardization bodies so that any healthcare institution with little effort can accurately understand their meanings. This effort is far less than coping with MLMs with various institutions each following their own standards. The standardization bodies have shared databases for their codes, which can be mapped easily by any institution to their local EMR for automatic translations. We have also added a definition of ‘events’ in our new MLMs using these standard codes.

3.3.2 Use of XML Representation

At present, the accepted representation of Arden Syntax is in text-based ASCII file format. HL7 Arden Syntax Special Interest Group also intends to move the text-based ASCII format to XML format. The group has agreed on shifting to XML format in four stages; however they have still yet to accomplish a big part of this task. We have created our own XML DTD ahead of them, which is described in Appendix B.

3.3.3 Conversion Tool

Since we have made changes to Arden Syntax, which includes the use of clinical standards and the adoption of XML format, we need some mechanism to convert existing, text-based ASCII MLMs to HealthWARNer format. For this purpose, one of our colleagues has developed an automated tool [47] to do this conversion.

3.3.4 Central Knowledge Base and MLM Processing Engine Sharing

We have used state-of-the-art technologies to design HealthWARNer, which has made HealthWARNer more superior than the earlier implementations of Arden Syntax. We will discuss the advantages that we have gained from our design in the later part of this section. We will begin with explaining an aspect of our design that involves the Central Knowledge Base and sharing of MLM processing engine (figure 3.2) whilst the other design aspects will be further explored in details in Chapter 6.

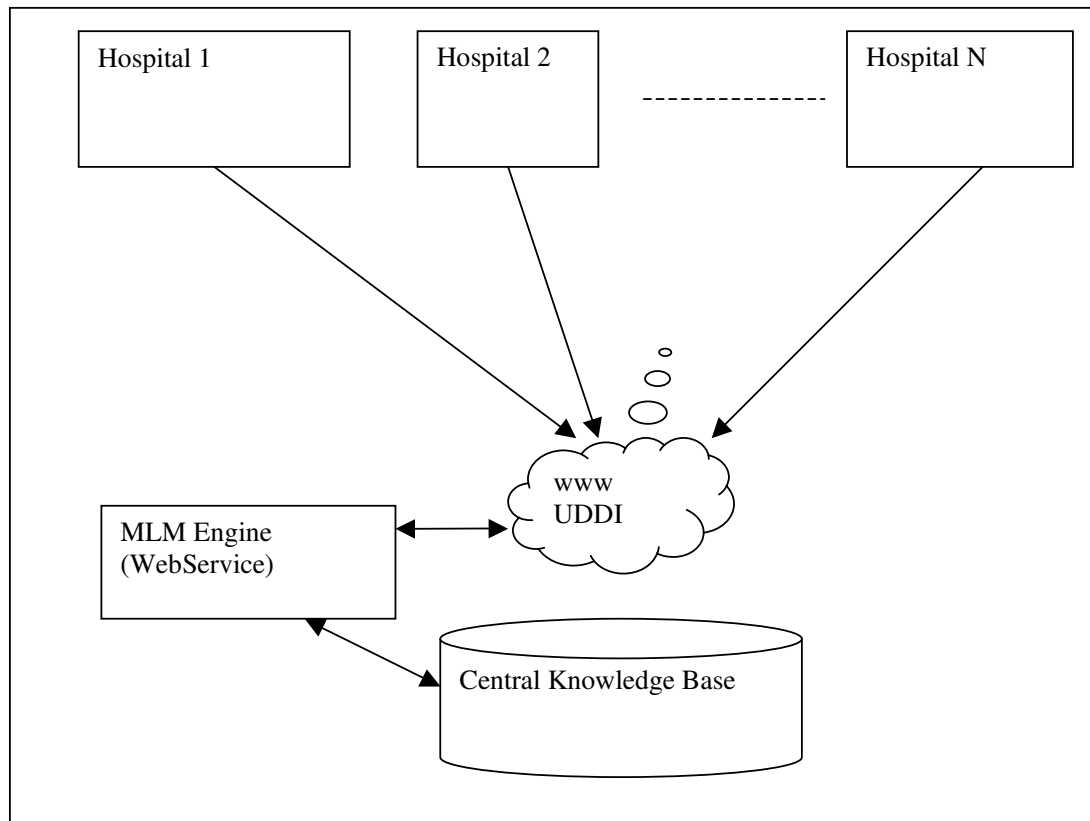


Figure 3.2 HealthWARNer Deployment

In HealthWARNer, MLMs are compiled locally within a healthcare enterprise but the runtime engine for processing the MLM knowledge can be located outside the healthcare enterprise environment and can be shared by a number of other healthcare institutions. This runtime engine is exposed as a WebService and can be discovered through a UDDI. HealthWARNer runtime environment also includes a Central Knowledge Base, which can act as a central repository for the knowledge from multiple healthcare institutions.

Since none of the Arden Syntax implementations was designed to share the runtime environment and the Central Knowledge Base amongst multiple institutions, it results in a number of issues that we need to first resolve. The first problem is the use of different clinical applications and platforms by the different healthcare institutions. To resolve this, we have used standard development environment such as JAVA and XML so that HealthWARNer's

built-time environment can fit into any platform and communicate with any other applications. Moreover, the HealthWARNer runtime application is using WebServices, which can be called from any platform using any application developed in any language. The second problem is caused by the use of different EMRs by different healthcare institutions. This issue cannot be resolved at the remote runtime environment because of the limitations of present day technology, so we have addressed this issue in the local built-time environment. In fact, this is the strongest reason for the need to have the built-time environment located at the healthcare institution. Details of how the built-time environment resolves this problem will be explained in section 6.2. The third problem is related to patient data privacy at the Central Knowledge Base and processing engine. We resolve this by separating the patient data from the knowledge and storing only the patient ID in the history. The fourth problem is the security of the knowledge at this central location. We have addressed this issue by introducing private and public security levels in the maintenance slot of a MLM. Using these constructs, the healthcare institution can restrict access to their own knowledge pool in the Central Knowledge Base to only certain authorized institutions.

After resolving the above-mentioned issues, it is possible to keep the knowledge in the central repository and share MLM processing engine amongst a number of healthcare institutions.

This gives us the following advantages:

1. It would become possible for the system to easily detect early outbreak of a disease or a biological attack.
2. Accelerate the discovery of new knowledge and improvement of existing knowledge by encouraging the idea of having a collective pool of MLM.
3. Save each institution the effort, time and finances to develop the same thing again.
4. Make it easy to adapt to changes in MLM syntax, which usually happens with the release of a new version. Using this approach the compiler would only be updated at a single location.

Chapter 4: Knowledge Creation, Evaluation and Improvement

As mentioned earlier, Arden Syntax lacked any mechanism to help improve the quality of knowledge residing in the MLMs. In order to improve the knowledge, the first step would be to have a mechanism to continuously measure the accuracy of the knowledge. We would discuss this mechanism provided by HealthWARNer in section 4.1 and 4.2. In Section 4.3 we will discuss the process through which new knowledge can be discovered, which we refer to as the Cycle of Knowledge Creation. This newly discovered knowledge is further refined by continuous measurement of accuracy and Cycle of Knowledge Creation. This cycle continues until all inaccurate knowledge is filtered out. Our research (section 2.2.2) showed the lack of interest displayed by the healthcare providers in adding knowledge in earlier Arden Syntax implementations. This proves to be a bottleneck in knowledge growth. In the last section of this chapter, we will discuss how HealthWARNer addresses this issue.

4.1 Addition of *Intention*, *Outcome* and *Event* to evaluate accuracy of knowledge

As knowledge accuracy calculation was absent in Arden Syntax, we have to introduce some new constructs to Arden Syntax. Two of these new slots are *intention* and *outcome* in the Maintenance slot in Arden Syntax. We would like to point out that MLMs already include a “purpose” slot, which may define the intention of the MLM as free text with no restriction on its content and structure. Intention slot is more effective than purpose slot as it can be

interpreted by computer program; Purpose slot is merely used to display the MLM purpose and could not be used to calculate efficiency automatically. The intention slot is structured and allows the software system to understand the purpose of the MLM. The outcome concept is entirely new to Arden Syntax and is essential for measuring efficiency of a MLM. Using these new constructs, the system can understand the intention and expected outcome; hence compare it with the patient's conditions to judge whether the knowledge in the MLM has indeed helped the patient.

We have provided various ways to describe intention and outcome of a MLM, described by the examples below:

```
A simplified portion of DTD for new intention and outcome slot:  
.....  
<!ELEMENT intention (warn_if | screen_for | check | .... )>  
<!ELEMENT warn_if (treatment_ordered, under_condition*)>  
<!ELEMENT treatment_ordered (treatment_standard_code)>  
<!ELEMENT screen_for (disease_standard_code | symptoms)>  
.....  
<!ELEMENT outcome ( (avoid |will_result_in), ( (event)*! (patient_state, time_frame) |  
    (event*, (patient_state, time_frame))) )>  
.....
```

Using the Document Type Definition (DTD) as illustrated above, MLM can have a structure intention. An example could be to *warn* or alert the doctor if he/she *orders a treatment* or prescribes a medication for a patient who has some specific *conditions*, under which the treatment or the medication can have adverse effects. This structure is computer interpretable.

```

Example of an intention slot:
<intention>
  <warn_if>
    <treatment_odered>
      .....
    </treatment_odered>

    <under_conditions>
      .....
    </under_conditions>
    <under_conditions>
      .....
    </under_conditions>
  </warn_if>
</intention>

```

Conditions can be drug allergy of any patient or some other fact, which is important but had been over-looked by the doctor while the prescription (*treatment_ordered*) was made.

An example of outcome can be to *avoid* a specific *patient state* that can be caused by that treatment *within a period of one week*.

```

An example of outcome slot:
<outcome>
  <avoid/>
  <event>
    .....
  </event>
  <patient_state>
    .....
  </patient_state>
  <time_frame>
    .....
  </time_frame>
</outcome>

```

We have introduced ‘time construct’ in outcome slot. According to the DTD, the outcome can be expected within a *time frame*, at the occurrence of an *event* or a combination of both. In the latter scenario, the system would evaluate the outcome when the event takes place or if the time elapses before that. The reason for this is that it is not always possible to specify a time frame for outcome to appear. A short estimated time frame might cause the system to check

the efficiency of the MLM before the patient's results are updated, resulting in evaluating incorrect results. A longer estimate would cause unnecessary delay. A better way can be to specify outcome as one or more events and check the efficiency of MLM immediately after that event takes place. The problem with this approach could be that it might not be possible to define an event in all cases or the event might never take place. To address these issues, we allow patient state and time frame to be specified either with an event or instead of an event. In order to cover all possible scenarios, we have left the option open to specify the desired patient state and the maximum time during which the system will wait for this specified patient state to appear.

4.2 Measuring Efficiency of Knowledge

Now that we know what the outcome would be and when that outcome will be seen, HealthWARNer can use this information to calculate the accuracy of a MLM. The Efficiency of a MLM is expressed through the following five rankings:

1. *Excellent*: If all the intended results were achieved within the specified time frame.
2. *Need improvement*: Expected output was seen but later than was expected
3. *Poor*: The expected patient state was not achieved
4. *Undetermined*: The outcome could not be judged or the treatment results were not recorded.
5. *Waiting for the results*: The maximum duration specified for the results to appear has not elapsed.

Here we must also add that due to the wide range of purposes of MLM, it is not possible to write an intention and outcome of each and every MLM in all the cases. Mainly these cases include where MLMs are being used for administrative purposes, rather than in clinical

decisions. For this purpose, we have made these new slots in MLM optional. This also makes our system backward compatible and helps in quick knowledge integration.

Each time the knowledge in an MLM is applied, its results would be accumulated as shown in Figure 4.1. This information can help in evaluating the performance of the knowledge in the MLM. Having this information for an old piece of knowledge and its new alternative treatment, can be used to compare their performances. Cases with their significant difference in the ratios of ‘Excellent’: ‘Need Improvement’: ‘Poor’ would indicate significant differences in the level of accuracy or effectiveness of those treatment plans.

Knowledge Ranking	Number of occurrences
Excellent	20
Need Improvements	5
Poor	2
Undetermined	3
Waiting for Results	6

Table 4.1: Accumulated knowledge ranking

Table 4.1 is summarizing the information described in MLM history (Figure 7.5). MLM History also provide link to Patient data, which could be used to further analyze a particular occurrence.

4.3 Cycle of Knowledge Creation

Clinical research is still far away from the stage where software systems can generate new knowledge entirely on its own and keep adding it to the system. Nonetheless, they can still help a lot in organizing, facilitating and automating some parts of the process of Knowledge creation. HealthWARNer introduces a new process to Arden Syntax for Knowledge creation and improvement, rather than the conventional approach adopted by earlier researchers to update the knowledge manually to the system. The earlier approach treats knowledge as a static component, while our approach is to treat knowledge as dynamic. HealthWARNer has the capability to automatically evaluate each piece of new knowledge to ensure that incorrect or inefficient knowledge is filtered out of the system. Figure 3.1 presents the high level view of this process called the Cycle of Knowledge Creation. It shows how new knowledge is being captured, evaluated and how our Active Knowledge Base (KB) absorbs this knowledge. It is a continuous process where a piece of knowledge created directly or created through this cycle is never assumed to be final and correct. MLM Knowledge passes through this cycle indefinitely for continuous refinement and improvement. This Cycle of Knowledge Creation has been built upon Arden Syntax, but it can be extended to other clinical support systems as well.

The above figure illustrates our process for knowledge creation. In the following sections, we will present an overview of major components and human roles involved in this system and then give a clinical example how knowledge is being improved by passing through this cycle.

4.3.1 The role of humans

There are three major roles defined in HealthWARNer's Cycle of Knowledge Creation

Process:

1. Knowledge Expert
2. Knowledge Translator
3. Knowledge Authenticator

Knowledge Expert

The first and the most important human role is that of the clinical knowledge expert, who usually is the practicing doctor. The model is based on the hypothesis that whenever a knowledge expert (practicing doctor) decides to ignore a treatment warning generated based on the knowledge of a MLM, then there is something that the he/she knows and is not there in the knowledge base. This could be an entirely new piece of knowledge or some facts, which are overlooked and have to be considered in that specific case. The main purpose of this model is to capture that knowledge and verify if it is useful or not. For this purpose, we have created a knowledge acquisition wizard (section 7.1) that interacts with the knowledge expert and prompt him/her with some structured questions and to capture his/her intention, expected outcome and logic for treatment.

Knowledge Translator

The Knowledge provided by this healthcare expert is in plain English and need to be translated to Arden Syntax. We have created a second role of 'knowledge translator' for this purpose. It is an interesting research problem to automate this part but as the project had

already spread quite a bit so we left this portion for future research. A benefit of having a translator is to make it easier for the practicing doctors to express their knowledge and experience in plain English with which they are more comfortable, while the translator could be proficient in XML and familiar with EMR. By using some good XML editing product, which graphically displays XML, the need to know XML format can be eliminated as well.

Knowledge Authenticator

Once the knowledge is translated to a MLM, we call it Crude Knowledge. This Crude Knowledge is not given the status of Active Knowledge but is used by the system to generate alerts and its efficiency is continuously calculated. The difference between these two categories of knowledge is that healthcare experts have authenticated ‘Active Knowledge’ while a doctor recommends ‘Crude Knowledge’ during treatment process. Once a MLM in Crude Knowledge gets the authentication, it is promoted to Active Knowledge, until then the alerts generated through this knowledge have an indication that they are not from Active Knowledge. Based on the Crude Knowledge efficiency calculations by the system, the third human role ‘knowledge authenticator’ decides whether the knowledge can be permanently added to the knowledge base as part of Active Knowledge. The MLM History (sections 7.2.6) gives good statistics about each MLM. Some threshold percentage can also be set to automate its approval or rejection. But as the system has not been tested, we find that it is safer for the time being to get an approval from the knowledge authenticator.

4.3.2 Computer-Human Interaction (Question and Answers)

We assume that when the doctors decide to *ignore* the system alerts against their prescribed treatment, then they are aware of some knowledge that is not part of the KB. The system will

interact with the Knowledge expert role to extract that knowledge. It is a challenging task to come up with a generic set of questions, which can extract relevant knowledge for any MLM, as they are very broad in their nature and purpose. For this purpose we have created a Knowledge Acquisition Wizard (section 7.1).

We have defined three broad sets of questions, which are related to intention, outcome and logic.

i. Intention related

The series of questions would start with the enquiry about the intention of the doctor. The first purpose of these sets of question is to find out whether the knowledge expert wants to add a new knowledge or improve on an existing one. If his/her intention is the same as that of an existing MLM (which generated the alert), he/she wants to improve existing knowledge and in the 'logic related' questions will be provided the same logic to modify. If his/her intention is different, the knowledge provided would be treated as a new piece of knowledge but first it would be confirmed from MLM history that the knowledge has not been tested and rejected earlier.

ii. Outcome related

The knowledge expert would be asked about his/her expected outcome in a similar way as the intention. He/she would also need to specify the expected patient state, which would appear after a specified time frame or a specified event. This would later be used for efficiency calculation of expressed knowledge.

iii. Logic related

The knowledge expert would need to explain his/her logic or methodology in plain English, which he/she used to reach his/her conclusion. He/she could also add citations to support

his/her decision. Here the logic varies in meaning from the logic slot in MLM. It could also include logic for evoke slot and action slot if they are different from the existing MLM.

How the Knowledge Acquisition Wizard posts these questions to the knowledge expert is explained further in section 7.1.

4.3.3 Active Knowledge

Active Knowledge is the part of the knowledge base which contains all the in-use authenticated MLMs.

4.3.4 Crude Knowledge

Once the knowledge translator translates the new knowledge proposed by the knowledge expert to Arden Syntax, it is stored in the Knowledge Base as Crude Knowledge. Crude Knowledge is treated slightly differently from the Active Knowledge. It is first tested for its validity and efficiency, and after the approval from the authenticator, it is moved to the Active Knowledge. We mark the MLM and the alerts generated from Crude Knowledge to ensure that it can be easily differentiated from Active Knowledge. It can still be applied to the patient under treatment as it has the recommendation of a practicing doctor.

4.3.5 Efficiency Measurement

This is a very important step in discovering new knowledge. The intention is HealthWARNer is not just to create knowledge but to continuously measure efficiency of new knowledge created to ensure that the knowledge created is accurate. This step of the process automates the process of filtering out the ineffective knowledge to make sure that the process only

produces useful knowledge. The efficiency of Crude Knowledge is measured based on the defined intention and outcome observed after the treatment. Effectiveness of a MLM is expressed through the following five rankings described in section 4.2

This system also measures the effectiveness of the Active Knowledge/in-use MLMs in the knowledge base if the intention and outcome are defined. If the doctor decides to apply the recommendation of a MLM, its efficiency is measured and stored in History.

4.3.6 MLM History

The results of the efficiency measurements are constantly updated in the history for each MLM as shown in figure 7.5. Having the rating for a MLM for a period of time is a quick indicator of its efficiency but the knowledge authenticator would need more information to make his/her decision on whether the knowledge is good enough to be placed in Active Knowledge. The information includes details of patient states before and after the treatment, the MLM logic and relevant patient history. Using all these and his/her expertise, he/she would conclude whether the MLM would prove beneficial for health care in the Knowledge Base or not.

Arden Syntax does not model the patient states so it is hard to extract this information. To overcome this problem, HealthWARNer generates two patient state reports. The first report is generated when the MLM is being executed; this will capture the patient state before the knowledge is applied. Queries are made to the EMR (Electronic Medical Record) to fetch patient data. There is a lot of patient data in the EMR, so there is a need to determine which data to fetch. We pick all the data referenced in the Logic and Evoke slot of the MLM. A MLM would essentially query all the relevant patient data to apply the knowledge so we would have all the necessary patient data in our queries. The second similar report is

generated when we check for MLM efficiency, which is when the outcome of treatment would appear. Together, these two sets of patient states can more clearly show the effect of the treatment specified in a MLM. In addition to this information, links to particular patient history and medical record are also provided to ensure that no other ill effects of the treatment was seen, allowing the knowledge authenticator to make a better judgment. Patient data is kept separate from the Knowledge by not storing it in the Knowledge Base along with MLM history. Only the patient ID is stored. To protect patient data privacy, the authenticator would require valid access rights to query patient data from the EMR.

In case a new knowledge is discarded, even then its history is retained for the review of knowledge experts and authenticators in future.

4.3.7 Knowledge Authentication

The knowledge authenticator receives the results of new knowledge efficiency and can view the MLM history. He/she has the authority to:

1. Accept the knowledge to be part of Active Knowledge.
2. Reject the knowledge. In this case the knowledge would be removed from Crude Knowledge but its history would be maintained to avoid future repetitions of same mistakes and it can also provide a base for future evaluation.

4.3.8 Example Scenario

To elaborate further on this Cycle of Knowledge Creation, we will present an example using a MLM from CPMC (Columbia-Presbyterian Medical Center -

<http://cs1xinfmtcs.csmc.edu/h17/arden>). We will make slight changes to it for the purpose of explaining the process.

MLM title:
“Screen for cytopenia in patients receiving cytopenic drugs (triggered by CBC then checks for cytopenic regimen by pharmacy order)”

Intention:
“Warn the health care provider of cytopenia in the setting of pharmacological therapy with cytopenics”

The expected output:
Clearer cytopenia symptoms would appear within two weeks time

Some explanation for its MLM:
“Whenever a CBC is stored in the database, the MLM checks for anemia, thrombocytopenia, and leukopenia and then checks for an active order for cytopenics. A warning is generated if the hemoglobin < (12.0 females or 13.5 males), the WBC is < 5.5, or the platelet count is < 150; and there is an active order for cytopenics. The warning is that that the patient may be experiencing cytopenia due to the pharmacologic agent known to cause cytopenia”

This MLM will be evoked when the event of ‘storage of CBC’ takes place, and if all the conditions specified in logic are fulfilled, an alert will be issued to the doctor who is treating the patient. Assume that the doctor who received the warning that the patient is experiencing cytopenia; according to his/her understanding, WBC level of 5.4 is acceptable for patients with such conditions. He/she decided to ignore this warning and carry on with his/her prescription.

The system will later ask him/her the questions regarding his/her intention, outcome and logic. In this case, his/her intention and outcome expected are the same as the existing knowledge. The change that he/she thinks is necessary is in the logic. So he/she would specify that this warning should have been delivered if the WBC level would have fallen below 4.5. The Knowledge translator would take the active MLM copy from the knowledge base, make the modifications in the logic and store it in the Crude Knowledge.

At this stage, some new knowledge is created but it has not been tested. After two weeks, the system would carry out the check for cytopenia in the patient record.

The positive or negative results of this would help the knowledge authenticators to decide whether they want to make the modifications in the piece of knowledge or not. Using the MLM history, the authenticators can see this patient's record before and after this alert was given. As the original MLM efficiency is constantly measured, they can also drill down to see the WBC level of other patients when this MLM sends advice to their doctor. If they see a general trend they could approve this modified knowledge that would give a more accurate alert in the future.

In a similar way, the knowledge expert could have suggested a missing condition, which is essential for identifying cytopenia in a patient. For example, he/she could have identified the missing hemoglobin level for males and females.

4.4 Improving Healthcare Provider Participation in Knowledge Creation

According to the conclusion of the survey in section 2.2.2, there is a general lack of interest on doctor's part in participating in knowledge gathering for similar systems. HealthWARNer is designed to encourage doctor's participation at many levels.

There are two ways in which this process encourages doctors to contribute to knowledge.

4.4.1 Cycle of Knowledge Creation

Make their task easy

HealthWARNer does not require doctors to be familiar with XML or Arden Syntax. Earlier Arden Syntax implementations required them to be familiar with Arden Syntax for MLM. Here, doctors do not have to waste their time and energy to learn and use these special skills to add knowledge in the knowledge base. We have a knowledge acquisition wizard that posts question and accepts answers in plain English and make it easy and quick for them to contribute their useful knowledge to the system.

Give them respect and importance

As indicated by earlier research, doctors generally are reluctant to contribute to knowledge. Therefore HealthWARNer Cycle of Knowledge Creation process takes the role of student and asks questions from the doctor, who is treated as a knowledge source. Doctor's knowledge and experience are given utmost importance and preference and hence would make them more willing to contribute to knowledge.

4.4.2 Audit of Treatment Process

Keeping record of treatments and medicines prescribed by a doctor for future audit of a patient is very common in clinical systems. The approach taken by HealthWARNer is slightly different and better. In general clinical systems, there is only one decision recorded which is the one the doctor made. There might be multiple decision alternatives but they all remain in the doctor's mind, only a single decision is recorded by the system. In HealthWARNer, besides the doctor, there might be one or more alternative provided by the knowledge in MLMs, which is like a second opinion. This alternative treatment suggested by HealthWARNer might be the same or different from doctor's decision from treatment. In case it is different and doctors have ignored HealthWARNer's opinion, it is recorded for the

purpose of any future audit. Coming back to the comparison with general clinical systems, here in the audited information, we would have an additional alternative treatment, which was ignored by the doctor. Having this more comprehensive audit information can lead to greater care taken by doctors when it comes to *ignoring* alerts generated by HealthWARNer without specifying any reasons.

Chapter 5: Healthcare Enterprise Knowledge

Integration

5.1 Use of Clinical Standards

As discussed in section 2.2.2, Curly braces were introduced in Arden Syntax to map MLM data with the institution's data. Most of the MLM need to query the patient data to make some decision. Arden Syntax did not define any standard structures for these queries and left it to the institution to define the query in whichever way they wish within the curly braces. The reason for this was to give institution the flexibility to use any type of database they wish. Though this managed to achieve some short-term objectives, but the 'curly braces problem' became a bottleneck in terms of reuse of MLM knowledge. This problem is cited by many publications as a limitation of Arden Syntax.

To resolve this problem, two issues need to be addressed here. Firstly, the structure/syntax needs to be made standard so that it can be read. Secondly, the terms/semantics need to be made a standard so that their meaning can be understood by the healthcare systems. In HealthWARNer, we have removed the curly braces from the MLM and replaced them with standard medical codes to ensure that any institution that wishes to integrate MLM knowledge into its healthcare enterprise system can do so without manual modifications.

There is no one standard which is comprehensive enough to describe all the various type of patient related data that can be possibly referenced within a MLM. Therefore we need to support a couple of standards in our MLMs. We have used Logical Observation Identifiers Names and Codes (LOINC)- <http://www.loinc.org> to represent laboratory observations,

International Statistical Classification of Diseases (ICD) and Related Health Problems - <http://www.who.int/whosis/icd10>) for diseases classification and National Drug Code (NDC)- <http://www.fda.gov/cder/ndc>) for drug codes. These are standardization bodies trying to classify and find standard codes so that any institution with little effort can accurately understand their meanings. This effort is far less than coping with MLMs with various institutions each following their own standards. The standardization bodies have shared databases for their codes, which can be mapped easily by any institution to their local EMR for automatic translations.

```
DTD example for data Representaions
.....
<!ELEMENT LIONIC (#PCDATA)>
<!ATTLIST LIONIC
    CODE CDATA #REQUIRED
    var_name ID #REQUIRED
    timestamp CDATA #REQUIRED>
<!ELEMENT NDC (#PCDATA)>
<!ELEMENT ICD (#PCDATA)>
<!ATTLIST ICD
    CODE CDATA #REQUIRED
    var_name ID #REQUIRED
.....
```

Another related problem that we noted was that Arden Syntax systems are event driven. There is no standard to define events, this leads to the problem where any institution that needs to adopt a MLM should somehow get to know what the event means preferably without human interaction. Expressing events in a standard way is challenging as they are treated in MLM as anything that happens to patient from the moment he/she enters the hospital till he/she leaves; for example 'Patient entering ICU', 'patient gets into cardiac arrest' etc. We can imply that events can be defined as a change in database. Not all terms in the institution database are standardized. So unless that is done there is no standard way to represent all the events. The

events related to patient diseases, laboratory test, and drugs prescribed which basically cover almost everything can be expressed using our own standard.

```
DTD example for defining events
.....
<!ELEMENT event ( ( LOINC, update) | (LOINC, insert) | (ICD, found) | (NDC, prescribed) | ....)>
.....
```

With the combination of xml tags (<update>, <insert>, <found> and <prescribed> etc.) we can use the existing standard code to represent events. For example, to specify the event: ‘glucose level stored’, we can use the glucose level LOINC code and <insert> tag. This will define an event, which can easily be understood with the help of “standard” defined by LOINC.

After using these standards, we find that it is much easier to integrate this knowledge to another Healthcare information system.

5.2 Arden Syntax XML Representation

5.2.1 Use of XML

HL7 Arden Syntax Special Interest Group discussed shifting Arden Syntax to XML format. They have defined four levels and plan to shift this standard from ASCII to XML representation of MLM in incremental stages. So far they have not completed the major portion of this transfer. We have defined our own XML DTD ahead of them for those portions, which they did not specify (Appendix B). Having an Arden Syntax represented in completely XML format would give it the added advantage of:

1. *Sharing HealthWARNer engine*: XML format is easily consumed by WebServices, making it possible for multiple institutions to share an engine for MLM processing.
2. *Simpler, more efficient and quicker development and adoption of changes in MLM*: An XML-tagged MLM makes it much more easier as compared to ASCII based text MLMs.
3. *Translation into other representations become simpler*: using XML style sheets
4. *Document Retrieval*: It will simplify the search specific MLMs from Knowledge bases. Moreover third party tools for XML search can be used.
5. *Transmission in XML-Based message Systems*: There is a great increase in use of XML-based messaging in healthcare. Having XML based MLM would make it easier to communicating with other systems.

5.2.2 Conversion Tool

As we have changed the representation of the Arden Syntax from text based ASCII files to XML, we need some tools to convert the existing text based ASCII MLM knowledge to XML format. There is no tool available for this conversion, so one of our colleagues has developed an automated tool to convert ASCII based MLMs to XML format [37]. Developing another similar tool to convert our XML based MLMs back to ASCII can also prove to be useful. We are also looking into developing this tool. Besides the reuse of knowledge, another advantage of having this conversion tool could be the reuse of existing editing tools [14] for ASCII MLMs for modeling XML based MLM.

Chapter 6: Centralized Knowledge Base and MLM

Processing Engine Sharing

As discussed in Chapter 5, HealthWARNer representation of MLM is much easier to share, consume and understand by another medical institution as compared to previous implementations of Arden Syntax. We have taken another step in sharing of knowledge by sharing the MLM Processing Engine and adding the concept of Central Knowledge Base. There are numerous benefits of that as discussed later in this chapter. In order to understand the Central Knowledge Base and MLM processing engine, we need to have some understanding of its architecture first. For this purpose, we have divided this chapter in two sections. The first section would discuss a high level view of the HealthWARNer design and some deployment strategies. This would show how the MLM Processing Engine and the Central Knowledge Base work. Later in the second section, we would go in details of why HealthWARNer supports Central MLM Processing Engine and Central Knowledge Base and what its benefits are. We would also describe the problems that we encountered in sharing the MLM Processing Engine and Central Knowledge Base and how we have addressed them.

6.1 Architecture

In the following sub-sections, we will highlight various components of HealthWARNer and describe the flexibility the system can demonstrate in its deployment.

High Level Design

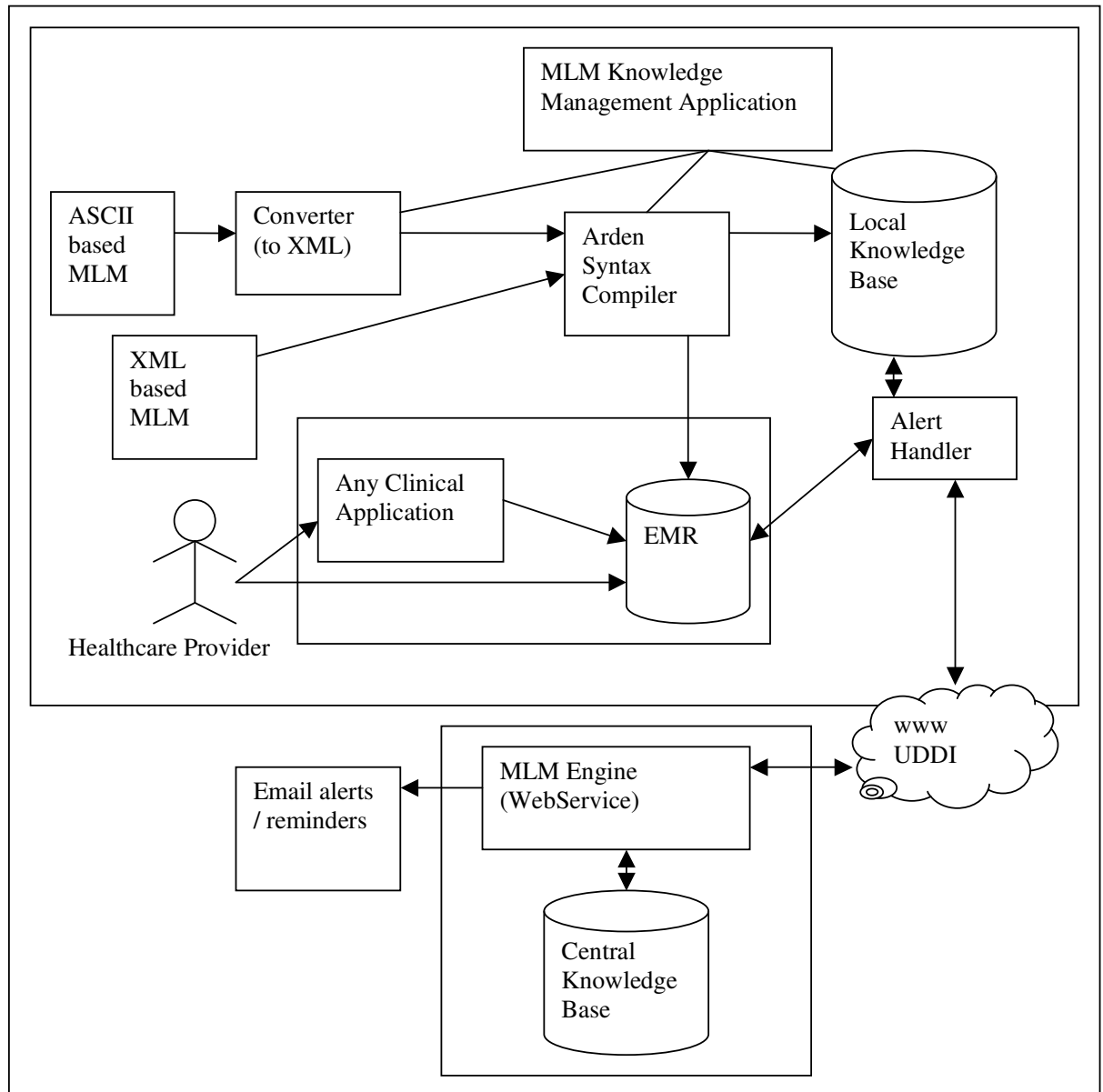


Figure 6.1: HealthWARNer Architecture

Figure 6.1 shows the architecture of HealthWARNer. In the following sub-sections we will describe each of these components to explain how MLM knowledge can be put in the system and used to generate alerts.

MLM

We support two types of MLM in HealthWARNer. One is our new XML based MLM (DTD shown in Appendix B). The second type is the text based ASCII MLM. We support this older version to make the following two reasons possible:

1. In case the users want to import knowledge from another institution, which is using older version.
2. Upgrade to HealthWARNer and still be able to use the earlier developed MLM

To use the second type of MLM, the user will use ‘MLM Knowledge Management Application’ to invoke the ‘converter’ [47], which will help to do the conversion with minimum human intervention. A small amount of human intervention is needed to do some mapping for the data slot.

Arden Syntax Compiler

Each MLM is compiled through this component before it is moved to the local knowledge base. This component does the validation and necessary entries in the database for future execution and triggering of the MLM. For each MLM it would create triggers in the EMR. A trigger is a database object, intelligent enough to detect modifications and insertion of patient data, for which we have a related MLM installed in the Knowledge Base. In the following subsections we would see that these triggers initiate the process of sending alert back to the Healthcare Providers.

Local Knowledge Base

The Local Knowledge Base stores the MLMs at healthcare enterprise level. Besides MLM knowledge regarding how to invoke a MLM, it contains stored procedures capable of invoking Java classes. These Java classes would notify the Alert Handler component that a MLM has been fired. These steps would take place when all the firing conditions of MLM are met. After explaining the Alert Handler, we will give an example scenario to explain how this local knowledge base interacts with the Alert Handler to notify that an event has taken place that will cause a MLM to fire.

Alert Handler

The Alert Handler component reads the detail information regarding how to fire a MLM from the local knowledge base. It is also capable of invoking the Webservice, which processes the remaining knowledge in the MLM and executes the action defined in it. A MLM can be specified to fire in a simple way, such as fire only once; or in a complex way, or it can be specified to fire periodically for a fix duration or time; or until another event takes place. Alert Handler will run as a demon and listen through RMI port for incoming messages from the local knowledge base. Once it receives a message, it starts its processing immediately. To further elaborate this, we present an example scenario:

An example scenario

A MLM has to fire every one hour from the time a patient enters ICU, until he/she leaves. In this case, triggers in the EMR will call the stored procedures in the Local knowledge base, when data regarding 'patient entering ICU' is inserted into EMR. The called stored procedures will inform Alert Handler through RMI and pass all the necessary information. The information includes the event: 'patient has entered ICU', the 'ID' of MLM that has fired and the 'ID' of the patient. Now the 'Alert Handler' will read Local knowledge base for further evoke condition of this particular MLM. The further information includes that the

MLM has to be invoked every one hour until the event ‘patient leave ICU’ takes place. After reading, it will keep on invoking the WebService to process the MLM every hour, until the event of ‘patient leaving ICU’ takes place.

MLM Knowledge Management Application

The Knowledge Management Application is used for the administration and knowledge management in HealthWARNer. It is a multi-user, role based web application, which can handle tasks ranging from adding MLM to knowledge base to supporting knowledge creation activities. This application will be explained in detail in section 7.2

Clinical Application & EMR

The first thing we would like to point out is that the component named ‘*any clinical application*’ in system diagram (figure 6.1) is not a component of HealthWARNer. We are using a demo clinical application and an EMR to demonstrate the capabilities of HealthWARNer. Any real world clinical application at the time of deployment would replace this demo application later. HealthWARNer is designed to easily integrate with any clinical application, which uses a database to store patient data, commonly referred as EMR. HealthWARNer creates triggers on the EMR rather than uses any part of the clinical application so it can generally work with any application. The database used should support trigger, which is a very common feature in commercial databases and is present in almost all of them. Even if triggers support is not present in the database, polling can be used instead to communicate with it, which can work even with the flat file databases.

MLM Engine (WebService)

MLM engine is a vital component of HealthWARNer and does the major part of MLM processing. The earlier components that we discussed so far handle only the evoke slot and

data slot of the MLM. This engine handles the most crucial parts of a MLM, which are the logic and the action slot. As we know, a MLM can make a single clinical decision; its logic slot holds the knowledge to make that decision. By processing this logic slot, the MLM Engine makes this decision. Once that decision is made, it takes the action specified in the action slot.

In figure 6.1 we have shown that the MLM Engine and the Central Knowledge Base can also be placed outside the boundary of a hospital, which is a better way to deploy HealthWARNer.

Central Knowledge Base

In situations where MLM Processing engine is being shared by a number of healthcare institutions as a collective effort to improve knowledge qualitatively and quantitatively, the Central Knowledge Base plays its role. It holds a central pool of MLMs, which can be shared amongst member healthcare institutions. It also supports private and public MLM concept to restrict and share MLM according to policies of each member. In Central Knowledge Base, data and knowledge are kept separate to protect patient data privacy. Data regarding MLM efficiency and MLM history is also stored here. This is a suitable place to deploy disease surveillance MLM, which can be used for an early detection of an outbreak of a disease or biological attack. Figure 6.2 shows an alert message generated by our sample MLM, which scans EMR for symptoms similar to Anthrax exposure.

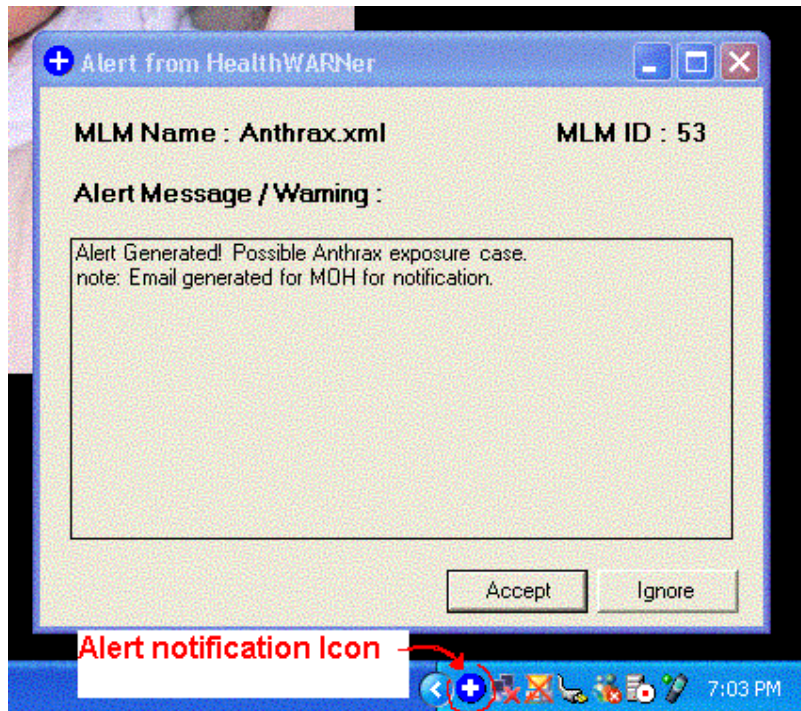


Figure 6.2: MLM Alert Notification Window

Deployment Strategies

HealthWARNer is a distributed system and can be very flexible in its deployment strategies. Depending on the need and the frequency of Alerts being fired, it can be installed on one to eight machines. Figure 6.3 shows the components highlighted in blue colour can be placed on separate machines. By using more machines, better load balancing can be achieved.

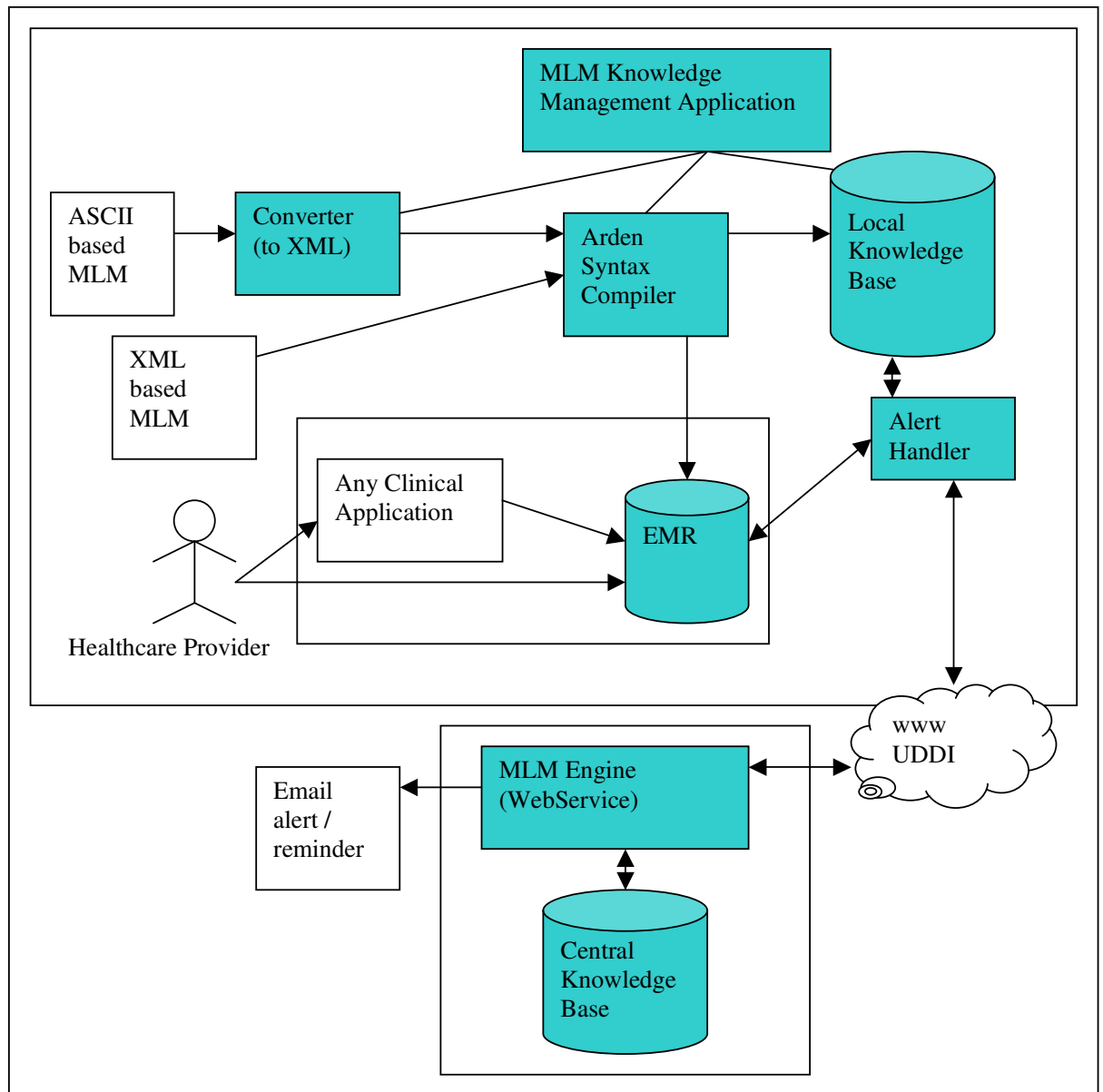


Figure 6.3: HealthWARNer Distributed Deployment

Figure 3.2 shows how the central MLM processing can be shared amongst a number of hospitals and clinics.

6.2 Centralized Knowledge Base and Process Sharing

Rationale and Benefits of Using WebServices

The current implementation models of Arden Syntax are standalone within a Healthcare Enterprise; they have no capability to interact with other similar systems. An institution, which is implementing such system, would create its own MLMs and has a local MLM processing engine, no collaboration or outside access is possible. All earlier research agrees to the importance of sharing MLMs but none has paid any attention to sharing the MLM processing engine. There are numerous advantages of doing so and these advantages are:

1. It would become possible for the system to easily detect early outbreak of a disease or a biological attack.
2. Accelerate the discovery of new knowledge and improvement of existing knowledge by encouraging the idea of having a collective pool of MLM.
3. Save each institution the effort, time and finances to develop the same thing again.
4. Make it easy to adapt to changes in MLM syntax, which usually happens with the release of a new version. Using this approach the compiler would only be updated at a single location.

Design Problems

Sharing the MLM processing engine is not that simple and has some problems that need to be addressed. These problems include:

1. Each institution might be using different clinical applications running on different platform.
2. Each institution might be using different EMR.
3. Patient data privacy.
4. Security or sharing of knowledge restrictions.

HealthWARNer is designed in a distributed fashion and hence addresses these issues in an effective manner. The solutions for each of these problems are outlined as follows:

Different Applications/Platforms

MLM processing engine is exposed as a WebService that makes it possible to be called from any platform using any application, which has been developed in any language.

Different EMR

The next problem is the use of different EMR by medical institution. This mainly affects the implementation of evoke slot of the MLM, which defines the condition for which the MLM is evoked. In HealthWARNer, this is kept local at the medical institution level, so this problem is also eliminated. There is a local 'Alert Handler' that needs to be installed in the institution which would process the evoke slot of the MLM and if all the conditions are met, the WebService would be invoked to process the MLM.

There are two other reasons to keep the 'Alert Handler' local and not part of the WebService. Firstly, Alerts are detected at first level using database triggers. Database triggers are implemented as database objects, which are generally required by most database providers to be locally installed. Secondly, in a typical Arden Syntax implementation, several events are taking place while patient data is entered and updated. Many of these events cause the evoke slot of MLM to be checked. When an evoke condition is checked, it is not necessary that

MLM would be further processed, it happens only when all the conditions are true. Practically only a fraction of events cause a MLM to be processed so it is not logical to invoke the WebService whenever an evoke conditions is checked for a MLM. So we do this processing locally to keep this load off the WebService and avoid expensive WebServices calls.

When we made this decision to keep evoke slot processing separate at institution level, it caused another problem. The problem was that we need the local ‘alert handler’ to be able to run on any platform and work with any database implementation. How we handled this problem was to make it a complete JAVA application. JAVA is platform independent and can be called through database triggers. Oracle databases allow JAVA classes to be added to the database, which can use RMI calls to notify the ‘Alert Handler’ about the occurrences of events. In the case of SQL server, triggers can call external executable files, which are Java classes capable of making RMI calls to ‘Alert Handler’. We have used SQL server as our database in the HealthWARNer prototype.

Patient Data Privacy

The next hurdle in having a ‘central MLM processor’ is patient data privacy. We have handled it by separating data from knowledge. Patient data is plugged in on the fly by the ‘alert handler’ before calling the WebService, which would use it for processing and not store it. MLMs are stored in the Central Knowledge Base with a patient ID only. Patient’s name, his symptoms and disease information are not stored. Patient ID is stored to help the Knowledge Authenticator to make a better decision for accepting or rejecting a MLM. The Knowledge Authenticator might get further information about the patient state before and after the MLM was used to apply some treatment to the patient. Patient ID can be used to fetch information from the EMR, provided the Knowledge Authenticator has the necessary access permissions. Using this method, we protect patient’s data privacy.

Security or Sharing of Knowledge Restrictions

Institutions, which invest their time and effort in creating MLM, might not want to share their MLM freely with other users of the 'central MLM processor'. To give them the choice to do so, we have added private/public MLM security level. This can be defined in the maintenance slot of MLM and would be used by the Central Knowledge Base to hide or show the MLM to other institutions.

So far, we are not using group-based sharing, but as the number of participating healthcare institutions increases, it would be required. Using this, MLM can be made public to a group of specified institutions while kept private to the rest of group(s).

Chapter 7: Implementation

In this chapter, we are going to discuss the implementation of HealthWARNer. We will start with describing some of the tools provided with HealthWARNer to manage knowledge and finally talk about the techniques and platforms used for HealthWARNer development

7.1 Knowledge Acquisition Wizard

As discussed in section 4.3, when the healthcare provider decides to ignore an alert generated by HealthWARNer and proceed with his/her own expert judgment, the system tries to capture this knowledge using the Knowledge Acquisition Wizard. This wizard will show up whenever the user decides to ignore an Alert generated by HealthWARNer by clicking *ignore* button in the Alert Window (see figure 6.2). The Knowledge Acquisition Wizard asks structured questions from the healthcare provider to document his/her purpose and logic in support of his/her judgment. This wizard allows the healthcare provider to update existing knowledge in the MLM or to add an entirely new piece of knowledge as shown in figure 7.1. In the case of update or insert, the existing knowledge is displayed to the healthcare provider for editing. This reduces his/her effort of typing and saves his/her valuable time. The healthcare provider does not have to write the MLM in XML format, but simply needs to express his/her logic behind the judgment in plain English. As mentioned in section 4.3, our hypothesis is that: *whenever the healthcare provider does not agree with the HealthWARNer, there is a piece of knowledge in his/her mind and not present in the knowledge base.* This wizard tries to capture this knowledge. Though there is an option provided by the wizard to ignore and not express the new knowledge, it is discouraged by the wizard with a warning message indicating that the action will be logged for any future audit, as seen in option selection in figure 7.1

In either case when the healthcare provider wants to update or add a new piece of knowledge he/she has to follow three steps. In step one, he/she is asked about his/her intention whether it is the same as the MLM, which sent the alert, or it is different. In step two, the wizard inquires about the expected outcome from his/her new knowledge. Finally in step three, he/she is asked to express his/her logic so that it can be put in the knowledge base as shown in figure 7.2. Once all the input has been made by the healthcare provider, all this information is posted to the knowledge translator's task list as shown in figure 7.6. Over there the translator will translate this knowledge expressed in plain English to an XML based MLM. This substantially relieves the healthcare provider from the effort to understand XML and save his/her time to express his/her knowledge in a MLM format.

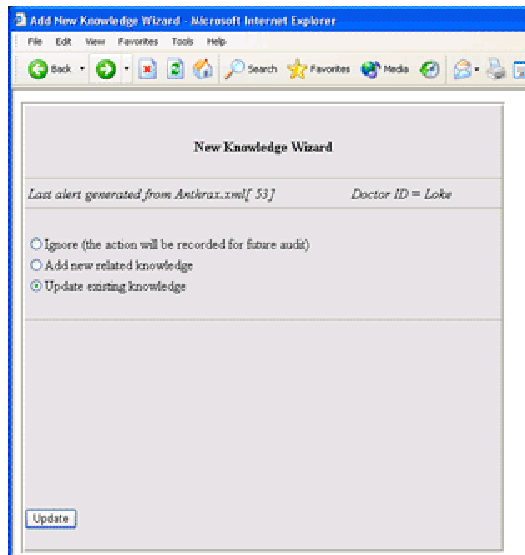


Figure 7.1: Knowledge Acquisition Wizard

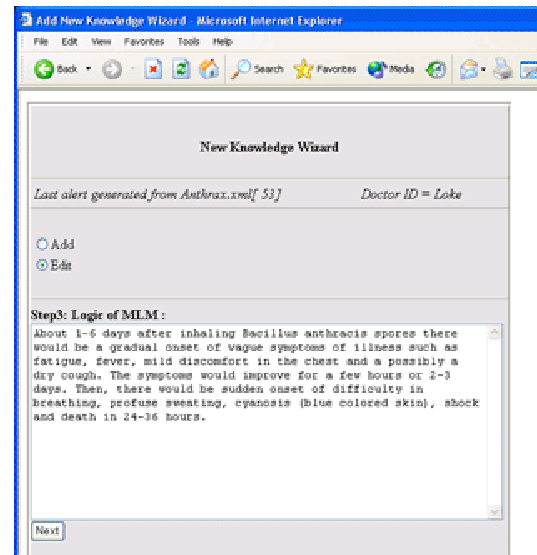


Figure 7.2: Specify new Logic

7.2 MLM Knowledge Management Application

MLM Knowledge Management Application is used to administer and manage the MLMs in the knowledge base and allow various users belonging to various roles to perform their knowledge management activities.

7.2.1 Roles

Users can belong to any one of the following roles:

1. Knowledge Authenticator.
2. Knowledge Translator
3. Administrator.
4. Healthcare Provider.
5. Knowledge Auditor.

7.2.2 Actions

The allowable actions are as follows:

1. Create New MLM
2. Modify MLM
3. Import MLM
4. Access MLM History
5. Translations
6. Users

7. Roles
8. Setting
9. Logout

As shown in table 7.1, the following default actions are allowed to existing roles:

		Roles				
		<i>Knowledge Authenticator</i>	<i>Knowledge Translator</i>	<i>Administrator</i>	<i>Healthcare Provider</i>	<i>Knowledge Auditor</i>
Actions	<i>New MLM</i>	.	Yes			
	<i>Edit MLM</i>	Yes	Yes			Yes
	<i>Import MLM</i>		Yes			
	<i>MLM History</i>	Yes			Yes	Yes
	<i>Translations</i>		Yes			
	<i>Users</i>			Yes		
	<i>Roles</i>			Yes		
	<i>Setting</i>	Yes	Yes	Yes	Yes	Yes
	<i>Logout</i>	Yes	Yes	Yes	Yes	Yes

Table 7.1: Roles and Allowed Actions

In the next sub-section, we will give a description of the actions allowed by the Knowledge Management Application.

7.2.3 Create New MLM

This option is used to add new MLM (figure 7.3) to the knowledge base. This is a simple to use option, which does a lot in the background. Once the user selects a MLM file from the hard disk, and clicks ‘Add MLM’ button, the MLM will be added to the knowledge base. In the background the xml file is first validated then Compiler is called which compiles and add MLM knowledge to the database. All the information about the evoke slot is kept in database

table for faster processing. Database triggers are created according to the evoke statement so that fast response is made to the clinical events. In case there is an error in the MLM file, it is displayed to the user; otherwise successful addition to knowledge base is also confirmed to the user.



Figure 7.3: Add new MLM

7.2.4 Modify MLM

This option is used for working with existing MLMs and can perform various actions on them and change their status. The available statuses are 'active' and 'disable'. Disabling a MLM will prevent it from giving any alerts. Here we must add that HealthWARNer does not allow the option to delete a MLM for audit purposes. This is because even if a MLM has not proved useful or was harmful, it can also be of value. It is kept in the history along with its outcome

so that the same mistake is not repeated. Besides this, we provide an option to edit the MLM here. This option points to the locations of the XML MLM file on the hard disk and opens it. There are numerous free XML-editing tools available on the Internet so we have not implemented another for HealthWARNer. A MLM is a standard XML file and any of the XML editors can be used to edit them.

Figure 7.4 shows the screen shot of the MLM editor. It shows a table in which a list of MLMs is given. User would have to select a MLM first and then choose one of the actions to be applied on it.

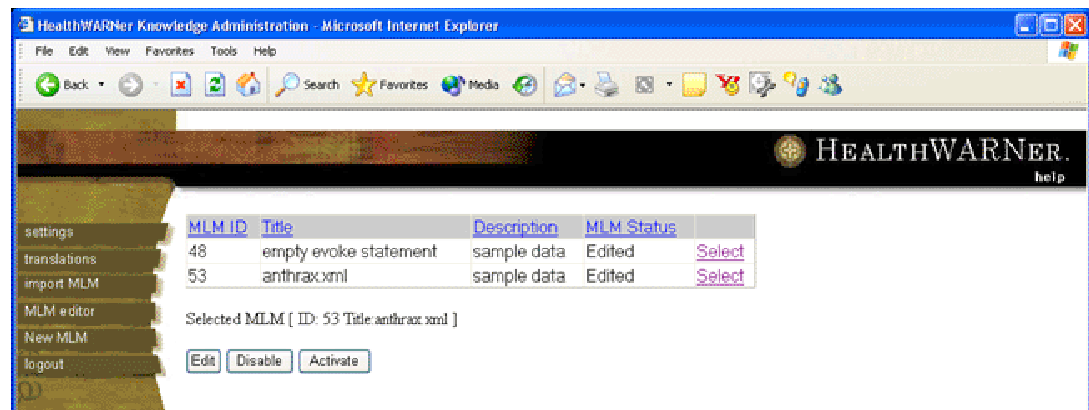


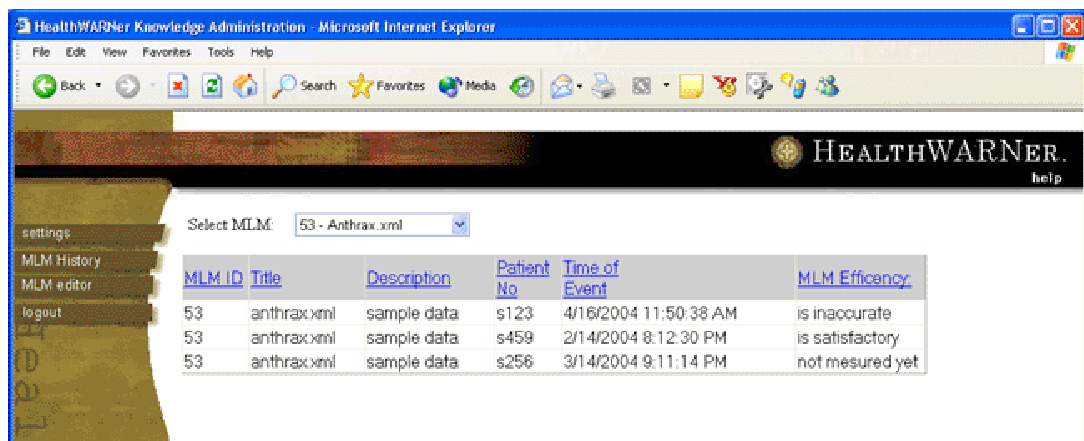
Figure 7.4: Edit MLM

7.2.5 Import MLM

This option is a link to one of our colleague's project [47] that wrote a web application to convert ASCII based MLM to XML based MLM. This option is the quickest way to convert text based ASCII MLM knowledge into this new format to get all the benefits provided by HealthWARNer.

7.2.6 MLM History

This is another essential part of knowledge improvement and creation features of HealthWARNer. User with the required privileges can select any MLM from the drop down menu shown in figure 7.5. Once a MLM is selected, all its history is shown in the table below. This table displays information about MLM performance each time it was executed. The information includes the efficiency of the MLM. The fourth column of the table shows the patient ID for which it was executed. Patient ID can be further drilled down to get information about the patient. The last column shows the measured efficiency of the MLM. In this particular case, MLM ID 53 proved accurate once, proved satisfactory the second time it was executed and the result for the third time has not been calculated yet. As discussed earlier, the results depend on the outcome slot of the MLM and usually would take some time for the patient to show results of the treatment; until then, the results are displayed as ‘not measured yet’. Once the results are out or the waiting time defined in the outcome slot expired, the efficiency value would change accordingly. For details of possible efficiency values, refer back to section 4.2.



The screenshot shows a web browser window titled "HealthWARNer Knowledge Administration - Microsoft Internet Explorer". The browser's address bar shows a URL starting with "http://". The page has a dark header with the "HEALTHWARNER." logo and a "help" link. On the left, there is a vertical navigation menu with links: "settings", "MLM History", "MLM editor", and "logout". The "MLM History" link is highlighted. The main content area features a "Select MLM:" dropdown menu with "53 - Anthrax.xml" selected. Below this is a table with the following data:

MLM ID	Title	Description	Patient No	Time of Event	MLM Efficacy
53	anthrax.xml	sample data	s123	4/16/2004 11:50:38 AM	is inaccurate
53	anthrax.xml	sample data	s459	2/14/2004 8:12:30 PM	is satisfactory
53	anthrax.xml	sample data	s256	3/14/2004 9:11:14 PM	not measured yet

Figure 7.5: MLM History

7.2.7 Translations

This option is part of the Cycle of Knowledge Creation process. According to the workflow, when the healthcare providers enter the knowledge in the Knowledge Acquisition Wizard, it is transferred to the task list of the translator. The translator will see the list of translation to be done in the 'pending task' table as shown in figure 7.6. Along with the pending translation task, some other useful information like MLM ID and the name of the doctor who created this piece of knowledge is also given. MLM ID can be very useful in case it is a modification to existing knowledge. It can be used to retrieve a copy of existing MLM for modification.

Doctor name can be used to verify any doubts in the mind of translator regarding the knowledge. Once 'Edit MLM' link is clicked in this table, the intention, outcome and logic as described by the doctor is displayed in the area below. As seen in figure 7.6, intention and outcome are still the same, while some additional conditions have been put in the logic slot of the MLM. So in this case the translator would have very little work to do, he/she would only update the logic slot in MLM ID 53 and press this update button to indicate that the MLM has been updated to the Crude Knowledge.

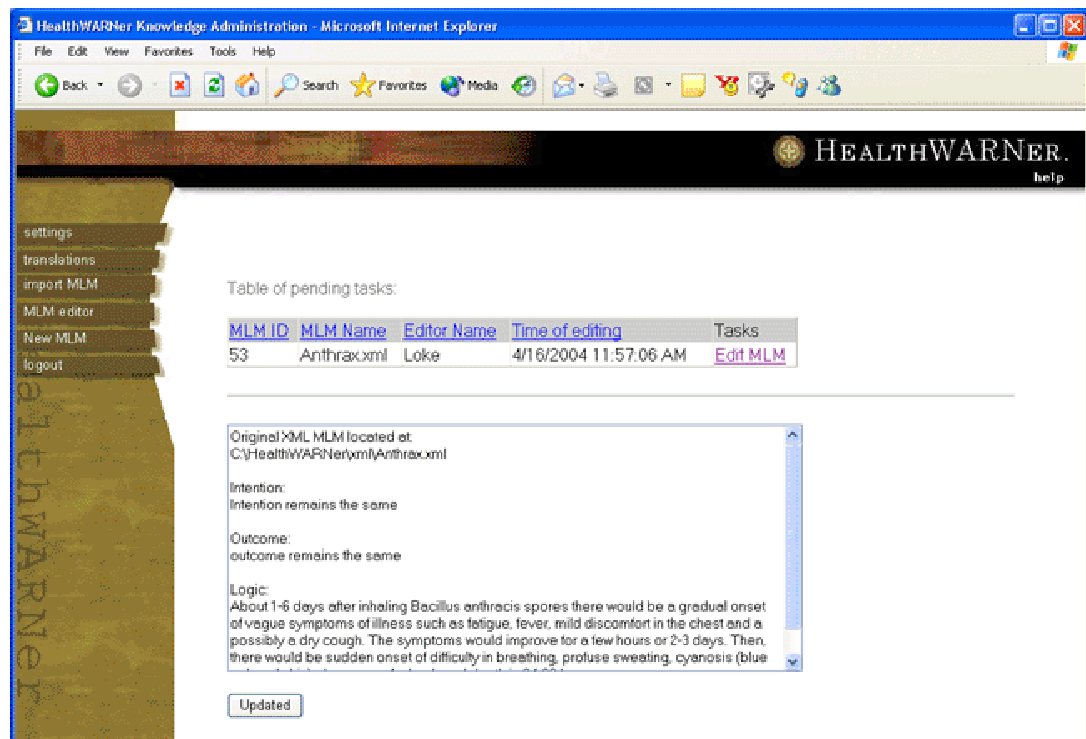


Figure 7.6: Translators Task List

7.2.8 Users

This option displays the existing users and their roles in MLM Knowledge Management Application. It can be also be used to create new users, delete existing ones, edit their personal information and reset password if needed. Figure 7.7 shows a screen shot of the user screen.

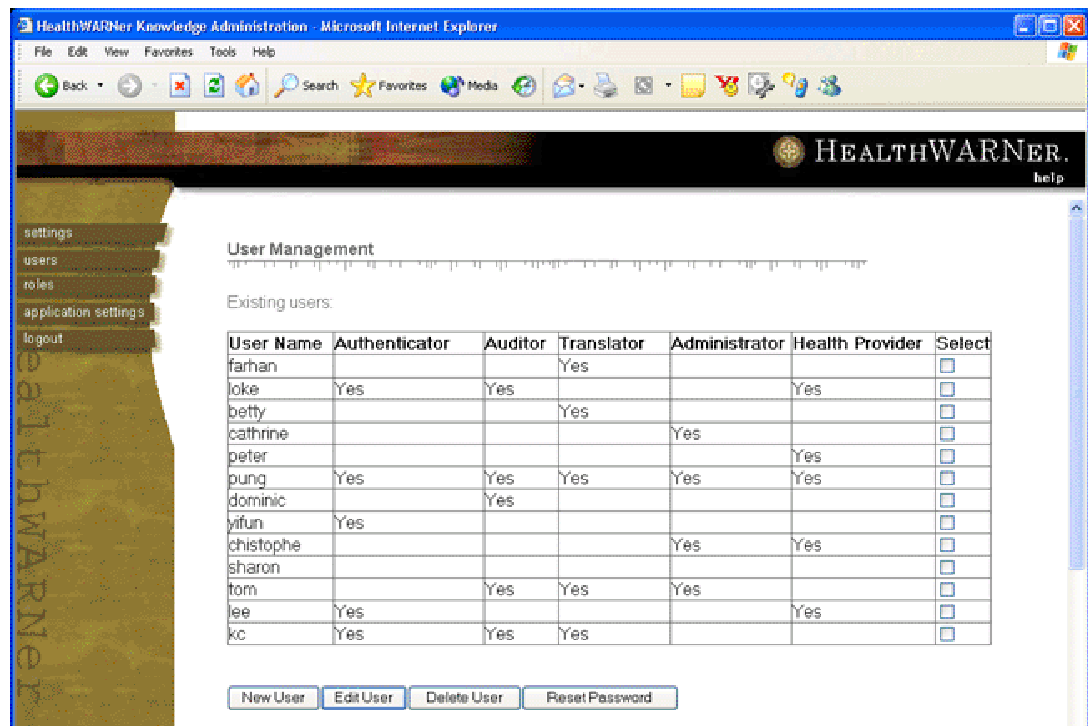


Figure 7.7: Users

7.2.9 Roles

An Administrator can use this option to assign roles (section 7.2.1) to user.

7.2.10 Setting and Logout

These are standard features for the purposes of customization and logging off from the session.

7.3 Tools

The design of HealthWARNer can be logically divided into two environments - the built-time and the run-time environment. The run-time environment processes the MLMs after they are fired. It also measures the efficiency of knowledge and maintains its history. Lastly, it facilitates the Cycle of Knowledge Creation process. The built-time environment allows the deployment of MLMs and supports knowledge management and administration features. We have developed the run-time environment, which is exposed as a WebService in .NET technology. Unlike run-time environment, which can be deployed externally, the built-time component needs to be installed within a healthcare institution, hence has to be able to run on any platform and communicate with any application written in any language. Therefore, we have used J2EE technology to design this built-time environment. We have used Microsoft SQL Server as a database for HealthWARNer prototype though our code is not strictly dependent on any particular database and can be extended to other commercial databases.

Chapter 8: Conclusion

In this chapter, we will evaluate HealthWARNer by discussing its performance and bottlenecks. We will further provide our test scenario to highlight its capabilities and strengths in detail. Thereafter, we will summarize the contributions of HealthWARNer towards clinical decision support system. Lastly, we will provide some pointers for future research in this area.

8.1 Evaluation

In order to evaluate HealthWARNer, we need to compare it with some existing system. The best candidate for this is the Arden Syntax implementation at CPMC. A study for that has been conducted at CPMC [35]. In the following sub-sections, we would talk about the system and functionality completeness, followed by performance evaluation and bottlenecks of this system. We would also present our test case details and results that we observed after creating a MLM to detect anthrax exposure in patients.

8.1.1 Structure of the System

Functionality Completeness

This is the strength of HealthWARNer, comparing the functionality of the HealthWARNer with the system implemented at CPMC. We find that features like the ones mentioned below are only present in HealthWARNer

1. Knowledge Efficiency Calculation
2. Cycle of Knowledge Creations
3. Central MLM processing

4. Use of Medical Standards
5. Completely 'XMLized'
6. Central Knowledge Base
7. Encourages Healthcare provider's participation

System Completeness

We have developed a working prototype of HealthWARNer with most of the features of Arden Syntax; however we have left out the implementation of some features, which are not essential to our research.

8.1.2 Performance

The Arden Syntax implementation at CPMC in terms of execution speed is better than HealthWARNer. Their study mentioned that their system should be able to handle 10 alerts per second during peak hours. HealthWARNer can handle a single alert in 5 seconds. Some performance slack could be attributed to the fact that during our test all its components and the EMR were running on a single machine (P III, 303MHz with 256MB of RAM). The system implemented at CPMC, on the other hand, is running on several machines. The specifications of those machines are not mentioned in the publication. Ignoring hardware capability, we did not expect HealthWARNer to be giving very high performance due to its two main bottlenecks. These two bottleneck processes are executed in sequence, which cause delay in alert processing.

Bottlenecks of HealthWARNer

i. Webservice Call

MLM processing is done at the central MLM Processing Engine, which is implemented as a WebService. WebServices are slow as they use http and they have standard protocols to follow, which require a lot of extra processing. That is a penalty in return for all the benefits of WebServices. We still use WebServices as their benefits outweigh this delay.

ii. XML Parsing

The second bottleneck is caused by the XML based MLM file parsing. These files are generally larger than text based ASCII MLM files. Writing code for processing XML file is faster as compared to text-based files but it takes more time to process. We are using DOM to parse the XML based MLM files, which as a first step, loads the whole MLM file in memory and validate it.

Comments

The processing of these two bottlenecks cannot be done in parallel as a WebService needs to be called first, before it can start processing the XML based MLM for that particular patient.

According to some feedback that we received for this system, it is generally perceived that our system response is slow due to the fact that we are continuously evaluating the efficiency of each MLM each time it is fired. This perception is inaccurate, as the efficiency is not calculated at the moment when alert is fired. Besides we are not relying on polling to check for the MLM efficiency results. For checking efficiency, an event is generated by the database and it is processed in a function in the database itself. The processing is very much optimized and does not slow down the response-time at all.

Clinical Testing

We understand that performance of features like the cycle of knowledge creation can be best tested in real clinical environment. Unfortunately we did not have any such environment to deploy and test HealthWARNer.

Testing in real clinical environment can get us a correct percentage of Healthcare providers who ignore HealthWARNer alerts and contribute their own knowledge to the system. More importantly we can find out the reasons that prevents healthcare providers from contributing new knowledge after they ignore an alert. This could possibly help us to further fine-tune the cycle of knowledge creation.

Furthermore we can find out the accuracy of the information extracted from the healthcare providers by our knowledge acquisition wizard. Describing the translated knowledge to its original contributor can help us do this. In case the generated knowledge is not exactly as the healthcare provider intended, we could possibly detect any mistakes or logical gaps in the question generated by the knowledge acquisition wizard.

8.2 Test Scenario

We have implemented a MLM called Anthrax.mlm to test the functionality and evaluation of HealthWARNer. In this section, we would share the results and our findings. We will start with presenting the piece of knowledge and how it is integrated to HealthWARNer. Then we

will proceed with how MLM is processed and alerts are generated from it, and finally, how knowledge is evaluated and discovered.

8.2.1 The Knowledge

The Raw knowledge in plain English is as follows:

“About 1-6 days after the inhalation of *Bacillus anthracis* spores, there would be a gradual onset of vague symptoms of illness such as fatigue, fever, mild discomfort in the chest and possibly a dry cough. The symptoms would improve for a few hours or 2-3 days. Then, there would be a sudden onset of difficulty in breathing, profuse sweating, cyanosis (blue colored skin), shock and death in 24-36 hours.”

The first step would be to write a MLM using this knowledge. As mentioned earlier, HealthWARNer uses advance representation of knowledge, which is in XML format as compared to predecessor Arden Syntax that was using ASCII base text file format. As this XML file is pretty large we have only placed important and relevant sections in Appendix A, example 2. The main benefits (discussed later) of this new representation come from its following new features:

1. All the symptoms are replaced with standard codes. For example ‘fatigue’ is replaced by ICD code 780.79, ‘difficulty in breathing’ is replaced by ICD Code 786.59.
2. Intention and outcome are put in place, which would be used to measure the MLM efficiency. In this case, it would be to correctly identify Anthrax exposure.

8.2.2 Knowledge Integration

Once we have this knowledge in XML format, it can be easily added within a few mouse clicks to HealthWARNer, as explained in section 7.2.3. In HealthWARNer, all the symptoms and diseases are replaced by standard codes; so it does not require any manual modification to

integrate this knowledge to HealthWARNer or any other system following its MLM representation as described in section 8.2.1. Mapping tables can be used to convert these standard codes to local EMR representation and vice versa.

8.2.3 Alert

We have a test clinical application to enter data in the EMR. Whenever the symptoms described in the knowledge (Anthrax.xml) are entered in the EMR through any clinical application, HealthWARNer will generate an Alert for the doctor that this might be an Anthrax case. We have modified the knowledge (action slot of MLM) to notify the Ministry of Health (MOH) of this possible Anthrax case by email as well.

8.2.4 Efficiency Measurement

As we have included intention and outcome in Antrax.mlm, which is to detect possible Anthrax case, HealthWARNer automatically starts waiting for the diagnosis of the particular patient for which it has generated this alert. If further tests on the patient conclude that it is anthrax exposure, HealthWARNer automatically updates its statistics for this particular MLM that this knowledge is accurate. New statistical results are added each time this MLM is fired. HealthWARNer is notified through the database trigger that it created on the EMR when the alert is fired. This trigger notifies HealthWARNer later about the accuracy of MLM diagnoses. As discussed in section 4.3.5, knowledge can have various levels of accuracy.

8.2.5 New Knowledge Discovery

As discussed in detail in section 4.3, we have a process called the Cycle of Knowledge Creation. It is very difficult to fully evaluate this step as it involves a human factor as well, which is the doctor's willingness to participate in this process. We can easily test the

infrastructure provided by HealthWARNer and the support it gives to the doctors in this knowledge discovery and management process. But it is hard to tell whether the steps that HealthWARNer would take to encourage doctors to participate in knowledge creation activity would be adequate or not (section 2.2.2). It would probably vary from individual to individual and could only be evaluated if this system is placed in a real test scenario where doctors use this system for advice while treating patients. Unfortunately we do not have the luxury of this test environment so we would just test the infrastructure provided by HealthWARNer.

When the alert window pops up on the doctor's computer as shown in figure 6.2, he/she is also provided with an option to ignore the alert if he/she thinks that a fact is being ignored by the MLM knowledge or perceived incorrectly. HealthWARNer attempts to capture this or an entirely new piece of knowledge in the following steps. Taking our present anthrax MLM knowledge, a doctor might disagree that this information is sufficient to make the diagnosis of Anthrax exposure and think that an additional blood test 'xyz' is needed to make a more accurate diagnosis. After pressing the ignore button (as discussed in section 7.1) the doctor would input this additional test requirement in plain English. The translator would add the LOINC code for this test in another copy of the same MLM and add it to the Crude Knowledge. Now these two MLMs would run independently and generate alerts for future patients with similar symptoms and record the outcome each time. Over a period of time, we would have statistics in the format as shown in figure 7.5 for both these MLMs. It should give a good idea about how many times each MLM diagnosis was accurate in predicting a possible Anthrax case. Based on the statistics, the Knowledge Auditor can decide to remove the incorrect or less useful MLM from the knowledge base. This is a continuous process and with time and discovery of better procedures, the knowledge base would automatically grow and improve.

8.3 Contributions

This thesis has made significant research contributions to Arden Syntax enhancement.

HealthWARNer, which is a more advanced Clinical Decision Support System, has extended the earlier work in several important aspects, as described in the following sub sections.

8.3.1 Knowledge Discovery Process

The Cycle of Knowledge Creation Process creates an environment where new knowledge is discovered and continuously refined. A concept like Central Knowledge Base further accelerates this knowledge creation process, as it opens the gateway for multiple institutions to participate in this process.

8.3.2 Knowledge Efficiency Calculation

HealthWARNer has provided a mechanism to constantly measure knowledge efficiency and keep the history of alerts generated from each MLM. This information can also help a lot in further research on the knowledge in a MLM.

8.3.3 Enterprise Knowledge Integration

Addition of medical standards and XML format to MLM has made them truly reusable. Now there is no manual modification required by any healthcare enterprise in order to share or consume MLMs from another healthcare enterprise. MLMs are now like a plug n play knowledge component, which can be easily integrated in the knowledge base with a few mouse clicks.

8.3.4 Easy to Adopt

HealthWARNer, with its central processing engine exposed as a WebService, Java components and standards reusable technologies can be easily integrated to any clinical application, written in any language and running on any platform. Compiler implementation by adopting institution is no longer required. The knowledge component having been significantly improved is now truly sharable and reusable making knowledge easy to adopt as well. Integration and deployment time and effort of this system into any environment is extremely lower than its ancestors.

8.3.5 XML based MLM

We have defined and implemented a DTD for XML based MLM, which has rendered numerous benefits to HealthWARNer.

8.3.6 Improved HealthCare Provider Participation

HealthWARNer is based on knowledge management principles, and encourages healthcare providers to participate in knowledge discovery process. Earlier implementation of such system did not pay much attention to this area.

8.3.7 Disease Surveillance

With features like Central Knowledge Base and WebService, this system has the capability of early detection of disease outbreak and biological attacks. It cannot prevent the unfortunate from happening but an early warning could help control and contain situation, in an effective way.

8.4 Future Research

Though we are quite satisfied with the research and finding of this thesis, we feel there is more that can be done in this area. HealthWARNer is based on a mature technology and we were concerned in the beginning that we would not find anything useful to work on, but we were proven to be wrong. This area is so wide and it has so much depth that very soon we had to start thinking which ideas to leave out. Some of them would be mentioned in this section for anyone who is interested in continuing work on this subject.

The Cycle of Knowledge Creation process that we discovered in our research can further be enhanced. The role of knowledge translator can be automated, which is a challenging project as the input is plain English (extremely unstructured and possibly incomplete) and the output is a MLM (structured, has to be complete). There would be information gaps, which would require to be filled. One possible way of doing that could be using hints from the existing base MLM. Automated verification of the accuracy of this knowledge before it can be placed in Knowledge base is also a challenging research project.

There are two major problems with Arden Syntax discussed by publications in this area. The first one is the curly braces problem, which we have addressed. The second one is the poor capability of Arden Syntax to handle complex guidelines [3, 13], which unfold over a longer period of time and can execute several parallel or concurrent plans in various orders. There has been an attempt to use intermediate states [2] to create care plans using Arden Syntax. A good aspect of the idea was that it used intermediate states for the flow and did not change the Arden Syntax at all. This made the implementation more easily adoptable as the institutions did not need to make changes in their compilers; however the technique was generally

considered awkward [12]. We too actually tried to address this issue using HISFLOW [25], which is a workflow for inter-organizational virtual healthcare enterprises. But we did not succeed in this as we were stretching HISFLOW too far beyond its intended capabilities. Finally, we had to give up on this idea, as it was unfeasible to modify HISFLOW and our projects had already spread beyond our initial intended scope. Besides we did not want it to address all the issue and provide unsatisfactory solutions, so we just dropped this one. But certainly workflows [24] can help here as shown by GUIDE (Petri net) system. There has to be some workflow out there that can handle single MLM as its activity and join them in a smooth fashion to create a complex network of knowledge. GridFlow might be the answer to this problem. This is again a huge and interesting problem, which can bring tremendous benefits to Arden Syntax standard.

8.5 Conclusion

This is a very interesting area of research and extremely useful as it can potentially help in saving human lives. We believe that we have made some useful contributions in the area of clinical decision support system by leveraging upon and further refining and extending Arden Syntax, a mature and widely accepted standard technology,

The design of HealthWARNer is guided by the motivation to provide a truly flexible, integrative, knowledge-absorbing system that can efficiently use the knowledge to prevent clinical errors. HealthWARNer is more than a receptacle that idly waits for the user to input and update the knowledge, like the earlier implementation of Arden Syntax. HealthWARNer constantly measures the efficiency of clinical knowledge that exists in the system. Its built-in process called Cycle of Knowledge Creation constantly tries to capture new knowledge from

the healthcare provider, and the knowledge moves through this cyclical process to get continuously refined. Unlike earlier systems, which treat knowledge in a static manner, HealthWARNer considers knowledge as a dynamic component of the system, and filters out treatment plans that are ineffective and inaccurate and replaces them with new and better treatment procedures. Involving the active participation of healthcare providers in the knowledge creation process is still one of the unexplored ways to model processes to discover new knowledge. HealthWARNer is the first clinical decision support system to have the appropriate measures and mechanism in place for this purpose.

The infrastructure of HealthWARNer is the synergy obtained from the combination of state-of-the-art-technologies like WebServices, XML, .NET and J2EE, which created an easy-to-adopt system that can virtually integrate with any platform running applications written in any language. None of the existing publications we have reviewed presents such a powerful clinical decision support system. Moreover, we have successfully incorporated clinical standards like LOINC, ICD and NDC to enhance Arden Syntax knowledge representation, making it truly integratable and reusable. Making knowledge fully integratable can greatly reduce the time and efforts required of the healthcare providers, which was previously spent on re-creating or modifying knowledge before it can be reused.

We have successfully made radical changes to Arden Syntax design to extend the scope of clinical decision support system from a single to multiple healthcare enterprises. Our knowledge processing engine and the knowledge itself are kept in a shared central location. This should greatly accelerate the process of knowledge discovery built in HealthWARNer. It would also enable timely or early detection of disease outbreaks and biological attacks.

The test cases discussed in this thesis to evaluate HealthWARNer indicated that we have basically achieved our objectives. However, we think that the real test for HealthWARNer

would be when it is placed in a real healthcare environment. The lessons learnt from such exercises would be extremely beneficial for further fine-tuning and future enhancements of HealthWARNer.

REFERENCES

- [1] The Knowledge-Creating Company. *Nonaka*. Harvard Business Review, November-December: 96-104.
- [2] Using intermediate states to improve the ability of the Arden Syntax to implement care plans and reuse knowledge. *Sherman EH, Hripcsak G, Starren J, Jenders RA, Clayton P*. Proc Annu Symp Comput Appl Med Care 1995;:238-42.
- [3] Moving Arden Syntax Outside of the (Alert) Box: A Paradigm for Supporting Multi-Step Clinical Protocols. *R. Matthew Sailors, ME, Richard L. Bradshaw, MS, Thomas D. East, Ph.D.* AMIA Annual Fall Symposium 1998: 1071.
- [4] EON: A Component-Based Approach to Automation of Protocol-Directed Therapy. *Mark A. Musen, Samson W. Tu, Amar K. Das, and Yuval Shahar*. Journal of the American Medical Information Association 3(6): 367-388, 1996.
- [5] Intention-Based Critiquing of Guideline-Oriented Medical Care: The Asgaard Project. *Aneel Advani, MD, MPH, Kinkoi Lo, MS, and Yuval Shahar, MD, PhD*. Proc AMIA Symp. 1998;:483-7.
- [6] Disseminating medical knowledge: the PROforma approach. *John Fox, Nicky Johns, Ali Rahmanzadeh*. Artificial Intelligence in Medicine, 14, 1998, 157-181.

- [7] Using Scenarios in Chronic Disease Management Guidelines for Primary Care. *Peter D. Johnson MB BS, Samson Tu M.S., Nick Booth MA MB BS MRCP DCH , Bob Sugden MBCS, Ian N. Purves MB BS, MRCP, MD.* Proc AMIA Symp. 2000;:389-93.
- [8] Protocols for Clinical Care. *Herbert, S.I., Gordon, C.J., Jackson-Smale, A., and Renaud Salis, J-L. (1995).* Computer Methods and Programs in Biomedicine, 48:21-6.
- [9] The GuideLine Interchange Format: A Model for Representing Guidelines. *Lucila Ohno-Machado, M.D., Ph.D., John H. Gennari, Ph.D., Shawn Murphy, M.D., Ph.D., Nilesch L. Jain, D.Sc., Samson W. Tu, M.S., Diane E. Oliver, M.D., Edward Pattison-Gordon, M.S. , Robert A. Greenes, M.D., Ph.D. , Edward H. Shortliffe, M.D., Ph.D. , G. Octo Barnett, M.D.* Journal of the American Medical Informatics Association 5(4):357-372, 1998.
- [10] GLIF3: The Evolution of a Guideline Representation Format. *Mor Peleg, Ph.D., Aziz A. Boxwala, M.B.B.S., Ph.D., Omolola Ogunyemi, Ph.D., Qing Zeng, Ph.D., Samson Tu, M.S., Ronilda Lacson, M.D., Elmer Bernstam, M.D., M.S.E., Nachman Ash, M.D., Peter Mork, B.A., Lucila Ohno-Machado, MD, Ph.D., Edward H. Shortliffe, M.D., Ph.D., Robert A. Greenes, M.D., Ph.D.* Proc. AMIA Annual Symposium, 2000.
- [11] GEM: A Proposal for a More Comprehensive Guideline Document Model Using XML. *Richard N. Shiffman, MD, MCIS, Bryant T. Karras, MD, Abha Agrawal, MD, Roland Chen, MD, LUIS Marengo, MD, Sujainnath, MD.* J Am Med Inform Assoc. 2000 Sep-Oct;7(5):488-98.
- [12] Sharable Representation of Clinical Guidelines in GLIF: Relationship to the Arden Syntax. *Mor Peleg, Ph.D., Aziz A. Boxwala, M.B.B.S., Ph.D., Elmer Bernstam, M.D., M.S.E. ,*

Samson Tu, M.S. , Robert A. Greenes, M.D., Ph.D., and Edward H. Shortliffe, M.D., Ph.D. J. Biomed Inform. 2001;34(3):170-181.

[13] Encoding a postoperative coronary artery bypass surgery care plan in the Arden Syntax. *Starren J, Hripcsak G, Jordan D, Allen B, Weissman C, Clayton PD. Comput Biol Med 1994;24(5):411 - 417.*

[14] MLM Builder: An Integrated Suite for Development and Maintenance of Arden Syntax Medical Logic Modules. *R. Matthew Sailors, ME Proc AMIA Annual Fall Symposium 1997:996.*

[15] Evolution of a Knowledge Base for a Clinical Decision Support System Encoded in the Arden Syntax. *Robert A. Jenders, MD, MS; Hao Huang, MPhil; George Hripcsak, MD; Paul D. Clayton, PhD. AMIA 1998 Annual Symposium.*

[16] Rationale for the Arden Syntax. *Hripcsak, G., Ludemann, P., Pryor, T. A., Wigertz, O. B., And Clayton, P. D. Computers in Biomedical Research, 27 (4), 291-324, 1994.*

[17] Promoting Workflow Integration with Information Management Services and GEM-Encoded Guidelines. *Abha Agarwal, MD, Cynthia A. Brandth, MD, MPH, Richard N. Shiffman, MD, MCIS. Proc AMIA Symp 2000.*

[18] Sharable Computer-Based Clinical Practice Guideline: Rationale, Obstacles, Approaches and Prospects. *Roberts A. Greenes, Mor Peleg, Aziz Boxwala, Samson Tu, Vimla Patel, Edward H. Shortliffe. Medinfo. 2001;10(Pt 1):201-5.*

- [19] The PROforma guideline specification language: progress and prospects. *Bury, J., Fox., Sutton, D.* Proceedings of the First European Workshop, Computer-based Support for Clinical Guidelines and Protocols (EWGLP 2000), Leipzig 13-14 Nov. 2000.
- [20] Asbru: A Task-Specific, Intension-Based, and Time-Oriented Language for representing Skeletal Plans. *Silvia Miksch, Yuval Shahar, Peter Johnsan.*
- [21] Is workflow Management Appropriate for Thaerapy Planning *Silvia. Miksch, Robert Kosara, Andreas Seyfang.*
- [22] Ontology-Based Configuration of problem-Solving Methods and Generation of Knowledge-Acquisition Tools: Application of PROTÉGÉ-II to protocaol-Based Decision support. *Samson W. Tu, Henrik Eriksson, John Gennari, Yuval Shahar, Mark A. Musen.* Artificial Intelligence in Medicine, 7, 201--225.
- [23] Approaches for Guideline Versioning Using GLIF. *Peleg M, Kantor R..* Proc AMIA Symp. 2003;:509-13.
- [24] Guideline-based careflow systems. *Silvana Quaglini, Mario Stefanelli, Andy Cavallini, Giuseppe Micieli, Clara Fassino, C. Mossa.* Artificial Intelligence in Medicine 20(1): 5-22 (2000).
- [25] Framework for distribution of activities of Interorganizational Workflows of a Virtual Health Enterprise. *Tauqir Amin, Pung Hung Keng.*
- [26] A Socio-Technical View of Knowledge-Sharing at Buckman Laboratories. *Pan, S.L., and Scarbrough, H.,*1998. Journal of Knowledge Management, Vol 2, No. 1.

- [27] Is US Health Really the Best in the World? *Barbara Starfield, MD, MPH*. JAMA July 26, 2000 Vol 284.
- [28] Implementing antibiotic practice guidelines through computer-assisted decision support: clinical and financial outcomes. *Pestotnik SL, Classen DC, Evans RS, Burke JP*. Ann Intern Med. 1996 May 15;124(10):884-90.
- [29] The Impact of Computerized Physician Order Entry on Medication Error Prevention. *David W. Bates, MD, MSc, Jonathan M. Teich, MD, PhD, Joshua Lee, MD, Diane Seger, RPh, Gilad J. Kuperman, MD, PhD, Nell Ma'Luf, Deborah Boyle and Lucian Leape, MD*. JAMIA : J Am Med Inform Assoc. 1999;6:313-321.
- [30] Visualization Techniques for Time-Oriented, Skeletal Plans in Medical Therapy Planning. *Robert Kosara*. Talk in the Konversatorium (colloquium) of the Institute of Computer Graphics and Algorithms of Vienna University of Technology, Vienna, Austria, June 1999.
- [31] AsbruView: Capturing Complex, Time-oriented Plans --- Beyond Flow-Charts Thinking with Diagrams. *Robert Kosara, Silvia Miksch, Yuval Shahar, Peter Johnson* an interdisciplinary workshop, Aberystwyth, UK, August 1998.
- [32] A User Interface for Executing Asbru Plans. *Robert Kosara Silvia Miksch*. Institute of Software Technology Vienna University of Technology, Austria.
- [33] Incidence of adverse drug reactions in hospitalized patients – A meta-analysis of prospective studies. *Lazarou J, Pomeranz BH and Corey PN*. JAMA 1998; 279(15):1200-5.

- [34] A Computer Alert System to Prevent Injury From Adverse Drug Events. *Raschke RA et al. JAMA* 1998, 280:1317-20.
- [35] Design of a Clinical Event Monitor. *George Hripcsak, Paul.D. Clayton, Robert A. Jenders, James J. Cimino, and Stephen B. Johnson. Computers and biomedical Research* 29, 194–221 (1996) ARTICLE NO. 0016.
- [36] Using Features of Arden Syntax with Object-Oriented Medical Data Models for Guideline Modeling. *Mor Peleg, Ph.D., Omolola Ogunyemi, Ph.D., Samson Tu, M.S., Aziz A. Boxwala, M.B.B.S., Ph.D., Qing Zeng, Ph.D., Robert A. Greenes, M.D., Ph.D., Edward H. Shortliffe, M.D., Ph.D. Proc AMIA Symp.* 2001;;523-7.
- [37] Representation of Clinical Practice Guidelines for Computer-Based Implementations. *Dongwen Wang, MPhila; Mor Peleg, PhD; Samson W. Tu, MSb;Edward H. Shortliffe, MD, PhDa; Robert A. Greenes, MD, PhDc. Medinfo.* 2001 ;10(Pt 1):285-9.
- [38] Protégé-2000: An Open-source Ontology-development and Knowledge-acquisition Environment. *Noy NF, Crubezy M, Ferguson RW et al. Proc AMIA Symp.* 2003;;953.
- [39] Death by Medicine. *Gary Null, Carolyn Dean and colleagues. Nutrition Institute of America,* October 2003.
- [40] The Syntax and Semantics of the PROforma guideline modelling language. *Sutton DR, Fox J. . J Am Med Inform Assoc.* 2003 Sep-Oct;10(5):433-43.

- [41] Combining diagnosis and treatment using ASBRU. *Andreas Seyfang, Silvia Miksch a, Mar Marcos b,c* (2002) MEDINFO 2001.
- [42] Using GEM-encoded guidelines to generate medical logic modules. *Agrawal A, Shiffman RN*. Proc AMIA Symp 2001;7-11.
- [43] Evaluation of guideline quality using GEM-Q. *Agrawal A, Shiffman RN*. Proceedings MEDINFO 2001; 1097-1101.
- [44] Non-Compliance with Guidelines: Motivations and Consequences in a case study. *Silvana Quaglini, Paolo Ciccarese, Giuseppe Micieli, Anna Cavallini*. Computer-based Support for Clinical Guidelines and Protocols, Proceedings CPG 2004 ,ed *IOS Press* ,pag. 75 - 87,(2004).
- [45] The NewGuide Project: guidelines, information sharing and Learning from Exceptions. *Paolo Ciccarese, Ezio Caffi, Lorenzo Boiocchi, Assaf Halevy, Silvana Quaglini, Anand Kumar, Mario Stefanelli*. Artificial Intelligence in Medicine, Proceedings AIME 2003 ,ed *Springer* ,pag. 163 - 167,(2003).
- [46] GLEE – a model-driven execution system for computer-based implementation of Clinical Practice Guideliness. *Wang D, Shortliffe EH*. Proc AMIA Symp. 2002;:855-9.
- [47] Text Based MLM to XML Based MLM Converter. *Law CY Jeffrey*. Thesis for Bachelor of Engineering, National University of Singapore.
- [48] Experiences with the Development, Implementation and Evaluation of Automated Decision Support Systems. *De Clercq PA, Hasman A*..To appear in Proc Medinfo 2004.

- [49] Design and implementation of a framework to support the development of clinical guidelines. *De Clercq PA, Blom JA, Hasman A, Korsten HHM*. Int J Med Inf 2001;64(2-3):285-318.
- [50] Supporting physicians in taking decisions in clinical guidelines: the GLARE "What if" facility. *Terenziani P et al*. AMIA 2002.
- [51] Executing clinical guidelines: temporal issues. *Terenziani P, Mastromonaco F, Molino G, Torchio M*. Proc AMIA Symp. 2000;;848-52
- [52] Telematics for Clinical Guidelines: A Conceptual Modelling Approach. *C Gordon, I Herbert, P Johnson, P Nicklin, P Reeves*. Proceedings of MIE'97.
- [53] Executing Clinical Practice Guidelines using the SAGE Execution Engine. *Ram P, Berg D, Tu SW et al*. To appear in Proc. Medinfo 2004.
- [54] DeGeL: A Hybrid, Multiple-Ontology Framework for Specification and Retrieval of Clinical Guidelines. *Shahar Y, Young O, Shalom E, Mayaffit A, Moskovitch R, Hessing A, and Galperin M*. Proceedings of the 9th Conference on Artificial Intelligence in Medicine—Europe (AIME) '03, Protaras, Cyprus, Oct. 2003, Springer-Verlag Heidelberg, pp. 122 - 131.
- [55] Round-Trip Engineering of Ontologies for Knowledge-Based Systems. Holger Knublauch and Thomas Rose. Twelfth International Conference on Software Engineering and Knowledge Engineering (SEKE), (239-247), Chicago, IL (2000)

[56] AsbruView: Visualization of Time-Oriented, Skeletal Plans. Silvia Miksch and Robert Kosara. The Fourth International Conference on Artificial Intelligence Planning Systems 1998

[57] Are Guidelines Following Guidelines?: The Methodological Quality of Clinical Practice. Shaneyfelt et al. *JAMA*. 1999; 281: 1900-1905.

[58] Representation Primitives, Process Models and Patient Data in Computer-Interpretable Clinical Practice Guidelines: A Literature Review of Guideline Representation Models. D. Wang, M. Peleg, S. W. Tu, A. A. Boxwala, R. A. Greenes, V. L. Patel, E. H. Shortliffe. *International Journal of Medical Informatics*, Vol. 68, No. 1-3, December 2002, p.59-70. 2002

[59] The quest for a computerized guideline standard: The process, its history, and an evaluation of the most common and promising methods used today. Matt Kavanagh and Susan Price MD. Oregon Health and Science University.

[60] An XML-based format for guideline interchange and execution. Dubey AK, Chueh HC. *Proc AMIA Symp.* 2000;:205-9.

[61] 1999 Institute of Medicine (IOM) Report

[62] Promoting Patient Safety: Overview of Clinical Decision Support. Robert Jenders, MD, John Dulcey, MD. AMIA 2001 Fall Symposium November 3, 2001

[63] Effects of computerized physician order entry and clinical decision support systems on medication safety: a systematic review. Kaushal R, Shojania KG, Bates DW. *Arch Intern Med.* 2003 Jun 23;163(12):1409-16.

Appendix A: MLM Examples

Example 1 ASCII Based text MLM

maintenance:

title: Alert on low hematocrit;;

filename: low_hematocrit;;

version: 1.00;;

institution: CPMC;;

author: George Hripcsak, M.D. (hripcsa@cucis.columbia.edu);;

specialist: ;;

date: 1993-10-31;;

validation: testing;;

library:

purpose:

Warn provider of new or worsening anemia.;;

explanation:

Whenever a blood count result is obtained, the hematocrit is checked to see whether it is

below 30 or at least 5 points below the previous value.;;

keywords: anemia; hematocrit;;

knowledge:

type: data-driven;;

data:

blood_count_storage := event

```

    {'complete blood count'};

hematocrit := read last

    {'hematocrit'};

previous_hct := read last {

    {'hematocrit'}

    where it occurred before the

    time of hematocrit};;

evoke: blood_count_storage;;

```

logic:

```

    if hematocrit is not number then

        conclude false;

    endif;

```

```

    if hematocrit <= previous_hct - 5

        or hematocrit < 30 then

            conclude true;

        endif;;

```

action:

```

    write "The patient's hematocrit      ("|| hematocrit ||") is low or
        falling rapidly.";;

```

end:

Example 2 Anthrax.mlm: XML Based MLM

```
<?xml version = '1.0'?>

<mlm_>

  <maintenance><title>Anthrax_surveillance01</title>

    <mlmname>Anthrax_surveillance01</mlmname>

    <arden><version>3</version></arden>

    <institution>National University of Singapore</institution>

    <author>Farhan Gul</author>

    <specialist><person><name>Farhan</name><degree>Bachelars</degree>

      <email>farhan@comp.nus.edu.sg</email></person></specialist>

    <date>25th Feb. 2003</date><validation><research/></validation>

    <security value="public"/></maintenance>

  <library><purpose>Give early warning in case Anthrax attack</purpose>

    <explanation>About 1-6 days after inhaling Bacillus anthracis spores there would be a
gradual onset of vague symptoms of illness such as fatigue, fever, mild discomfort in the
chest and a possibly a dry cough. The symptoms would improve for a few hours or 2-3 days.
Then, there would be sudden onset of difficulty in breathing, profuse sweating, cyanosis (blue
colored skin), shock and death in 24-36 hours.if conditions B are true and presence of
conditions A, send email to MOH and alert msg to doctor</explanation>

    <keywords><keyword>Anthrax</keyword><keyword>Disease surveillance</keyword>

      <keyword>Bio Attack</keyword></keywords>

    <links><li type="url_link">http://www.emergency.com/anthrax2.htm</li></links>

    <intention><screen_for><disease_standard_code>022.9</disease_standard_code>

      </screen_for></intention>

  <outcome><will_result_in/><event timestamp=""><ICD CODE="022.9" var_name="b"
timestamp="a">true</ICD><found/></event><patient_state/><time_frame>
```

```

<evoked_duration><number><digits>2</digits></number><duration_op
value="DAYS"/></evoked_duration></time_frame></outcome>

.....

  <data_slot><data_block><data_statement><data_assignment><ICD CODE="022.9"
var_name="anthrax_presence" timestamp=""/><ICD CODE="780.79"
var_name="fatigue" timestamp="2004-04-14 20:05:03.127">true</ICD><ICD
CODE="780.6" var_name="fever" timestamp="2004-04-14 20:05:03.127">true</ICD>
<ICD CODE="786.59" var_name="chest_discomfort" timestamp="2004-04-14
20:05:03.127">true</ICD></data_assignment>

    <event_assignment><event timestamp="2003/09/14 21:11:05:1000">
<ICD CODE="786.0" var_name="difficulty_in_breathing" timestamp="2004-04-14
20:05:03.127">true</ICD><found/></event><event timestamp="2003/09/14
21:11:05:1000"><ICD CODE="780.8" var_name="profuse_sweating" timestamp="2004-04-
14 20:05:03.127">true</ICD><found/>
.....

<priority>10</priority>

<evoked_slot><evoked_block><evoked_statement><event_or><event_any>

<identifier value=" " ref="difficulty_in_breathing" time=""/><identifier value=" "
ref="profuse_sweating" time=""/><identifier value=" " ref="cyanosis" time=""/>
</event_any></event_or>

.....

  <logic_slot><logic_block><logic_statement>

    <IF/><logic_if_then_else2><expr><expr_sort><expr_where>

.....

<identifier value="true" ref="fatigue" time=""/></expr_factor_atom>
</expr_factor></expr_function></expr_function></expr_ago></expr_before>
<is/><main_comp_op><temporal_comp_op><WITHIN/><PAST/>

```


.....

<number><digits>**8**</digits></number></expr_factor_atom></expr_factor>
</expr_function><duration_op value="**DAYS**"></expr_duration></expr_ago>

.....

<**CONCLUDE**><boolean_value value="**true**"></CONCLUDE></logic_statement>
</logic_block><logic_elseif><**ELSE**></logic_block><logic_statement><**CONCLUDE**>
<boolean_value value="**false**"></CONCLUDE></logic_statement></logic_block>

.....

</logic_block></logic_slot>
<**action_slot**><action_block><action_statement><**email**>**farhan@comp.nus.edu.sg**</email
><**institution**>**National University Hospital**</institution><string>**Patient arrived with
possible Anthrax symtoms**</string></action_statement><action_statement>
<message>**Caution: Possible anthrax effected patient**</message></action_statement>
</action_block> </action_slot><**urgency**>**1**</urgency></knowledge></mlm_>

Appendix B: DTD for XML based MLM

```
<!ELEMENT mlm_ ( maintenance, library, knowledge )>

<!ELEMENT maintenance ( title, mlmname, arden, institution, author,
specialist, date, validation, security )>

<!ELEMENT library ( purpose, explanation, keywords?, citations?, links?, intention,
outcome)>

<!ELEMENT knowledge ( type, data_slot, priority?, evoke_slot, logic_slot,
action_slot, urgency? )>

<!ELEMENT title (#PCDATA)>

<!ELEMENT mlmname (#PCDATA)>

<!ELEMENT arden (version)>

<!ELEMENT version (#PCDATA)>

<!ELEMENT institution (#PCDATA)>

<!ELEMENT author (#PCDATA)>

<!ELEMENT specialist (person)>

<!ELEMENT date (#PCDATA)>

<!ELEMENT validation (production|research|testing|expired)>

<!ELEMENT production EMPTY>

<!ELEMENT research EMPTY>

<!ELEMENT testing EMPTY>

<!ELEMENT expired EMPTY>

<!ELEMENT person ( name, degree*, email )>

<!ELEMENT name (#PCDATA)>
```

```

<!ELEMENT degree (#PCDATA)>

<!ELEMENT email (#PCDATA)>

<!ELEMENT security EMPTY>

<!ATTLIST security  value (private|public) #REQUIRED>

<!ELEMENT purpose (#PCDATA)>

<!ELEMENT explanation (#PCDATA)>

<!ELEMENT keywords (keyword*)>

<!ELEMENT keyword (#PCDATA)>

<!ELEMENT citations (ci*)>

<!ELEMENT ci (#PCDATA)>

<!ATTLIST ci ci.number CDATA #IMPLIED

           ci.level (support|refute) #IMPLIED>

<!ELEMENT links (li*)>

<!ELEMENT li (#PCDATA)>

<!ATTLIST li

           type (url_link|mesh_link|other_link|exe_link) #IMPLIED  >

<!ELEMENT intention (warn_if | screen_for | check )>

<!ELEMENT warn_if (treatment_ordered, under_condition*)>

<!ELEMENT treatment_ordered (treatment_standard_code)>

<!ELEMENT screen_for (disease_standard_code | symptoms)>

<!ELEMENT disease_standard_code (#PCDATA)>

<!ELEMENT outcome ( (avoid | will_result_in), ((event)*|(patient_state,time_frame)

| (event*, (patient_state, time_frame)) ) )>

<!ELEMENT will_result_in EMPTY>

<!ELEMENT patient_state (#PCDATA)>

```

```

<!ELEMENT time_frame (evoke_duration)>

<!ELEMENT type (data_driven)>

<!ELEMENT data_driven EMPTY>

<!ELEMENT data_slot (data_block)>

<!ELEMENT priority (#PCDATA)>

<!ELEMENT evoke_slot (evoke_block)>

<!ELEMENT logic_slot (logic_block)>

<!ELEMENT action_slot (action_block)>

<!ELEMENT urgency (#PCDATA)>

<!ELEMENT logic_block (logic_statement+)>

<!ELEMENT logic_statement (empty |logic_assignment|
(IF, logic_if_then_else2) |(FOR, identifier, IN, expr, DO, logic_block, ENDDO)|
(WHILE, expr, DO, logic_block, ENDDO)| (CONCLUDE, expr)| (CONCLUDE) )>

<!ELEMENT IF EMPTY>

<!ELEMENT logic_if_then_else2 (expr, THEN, logic_block, logic_elseif )>

<!ELEMENT logic_elseif (ENDIF | (ELSE, logic_block, ENDIF) |
(ELSEIF, logic_if_then_else2)))>

<!ELEMENT logic_assignment ( (identifier_becomes, expr) | (time_becomes, expr)|
(identifier_becomes, call_phrase) )>

<!ELEMENT identifier_becomes ( identifier, equal_to)>

<!ELEMENT time_becomes (time_of, identifier, equal_to)>

<!ELEMENT call_phrase ( (CALL, identifier) | (CALL, identifier, WITH, expr) )>

<!ELEMENT CONCLUDE (boolean_value)>

```

```

<!ELEMENT expr ( expr_sort | (expr, expr_sort) )>

<!ELEMENT expr_sort ( expr_where | (expr_where, merge, expr_sort) | (sort,
expr_sort) | (sort, sort_option, expr_sort) )>

<!ELEMENT where EMPTY>

<!ELEMENT sort EMPTY>

<!ELEMENT sort_option EMPTY >

<!ATTLIST sort_option
        value (TIME|DATE) #REQUIRED>

<!ELEMENT expr_where (expr_range | (expr_range, where, expr_range) ) >

<!ELEMENT expr_range (expr_or | (expr_or, seqto, expr_or) )>

<!ELEMENT seqto EMPTY >

<!ELEMENT expr_or ( (expr_or, or, expr_and) | expr_and) >

<!ELEMENT or EMPTY>

<!ELEMENT expr_and ((expr_and, or, expr_not) | (expr_and, and, expr_not) |
expr_not ) >

<!ELEMENT expr_not ((not, expr_comparison) | expr_comparison) >

<!ELEMENT expr_comparison (expr_string | (expr_string, is, main_comp_op) |
(expr_string, simple_comp_op, expr_string) |
        (expr_string, is, not, main_comp_op) |      (expr_string, occur,
temporal_comp_op)|
        (expr_string, occur, not, temporal_comp_op)+ | (expr_string,
occur,range_comp_operator) |
        (expr_string, occur, not, range_comp_oerator) | (expr_string, matches,
pattern,expr_string) ) >

<!ELEMENT not EMPTY>

```

<!ELEMENT is EMPTY>
 <!ELEMENT occur EMPTY>
 <!ELEMENT matches EMPTY>
 <!ELEMENT pattern EMPTY>
 <!ELEMENT expr_string (expr_plus | (expr_string, double-pipe, expr_plus) |
 (expr_plus, formatted, with, format_string))>
 <!ELEMENT double_pipe EMPTY>
 <!ELEMENT formatted EMPTY>
 <!ELEMENT with EMPTY>
 <!ELEMENT format_string (format_specification)>
 <!ELEMENT format_specification ((format_specification,
 format_specification_single)| format_specification_single)>
 <!ELEMENT format_specification_single ((format_options, format_flag, width,
 precision) | text)>
 <!ELEMENT format_options EMPTY>
 <!ATTLIST format_options
 value (positive|negative|zero|space|hash) #REQUIRED>
 <!ELEMENT format_flag EMPTY>
 <!ATTLIST format_flag
 value (c|C|d|D|e|E|f|F|g|G|h|I|l|L|n|p|s|t) #REQUIRED>
 <!ELEMENT width (empty | digits)>
 <!ELEMENT precision (empty | digits)>
 <!ELEMENT expr_plus (expr_times | (expr_plus, plus, expr_times) |
 (expr_plus,
 minus, expr_times) | (positive, expr_times) | (negative, expr_times))>

```

<!ELEMENT expr_times (expr_power | (expr_times, multiply, expr_power) |
(expr_times, divide, expr_power) )>

<!ELEMENT expr_power (expr_before | (expr_function, raised_to_power_of,
expr_function) )>

<!ELEMENT expr_before (expr_ago | (expr_duration, BEFORE, expr_ago) |
(expr_duration, AFTER, expr_ago) | (expr_duration, FROM, expr_ago ) )>

<!ELEMENT expr_ago (expr_function | expr_duration | (expr_duration, AGO))>

<!ELEMENT expr_duration (expr_function, duration_op)>

<!ELEMENT expr_function (expr_factor | (of_func_op, expr_function) |
(of_func_op, OF, expr_function) | (from_of_func_op, expr_function) |
(from_of_func_op, OF, expr_function) |
      (from_of_func_op, expr_factor, FROM, expr_function) | (from_func_op,
expr_factor, FROM, expr_function) | (index_from_of_func_op, expr_function) |
(index_from_of_func_op, OF, expr_function) |
      (index_from_of_func_op, expr_factor, FROM, expr_function) |
(index_from_func_op, expr_factor, FROM, expr_function) | (expr_factor, AS,
as_func_op) )>

<!ELEMENT expr_factor (expr_factor_atom | (expr_factor_atom, expr) )>

<!ELEMENT expr_factor_atom (identifier | number | string | time_value |
boolean_value | NULL | it | empty_brackets | expr )>

<!ELEMENT it (IT | THEY)>

<!ELEMENT simple_comp_op EMPTY>

<!ATTLIST simple_comp_op
      value (EQ|LT|GT|LE|GE|NE) #REQUIRED>

```

```

<!ELEMENT main_comp_op (temporal_comp_op | range_comp_op |
unary_comp_op |
(binary_comp_op, expr_string) )>

<!ELEMENT range_comp_op (WITHIN, expr_string, TO, expr_string )>

<!ELEMENT temporal_comp_op ( (WITHIN, expr_string, PRECEDING,
expr_string) |
(WITHIN, expr_string, FOLLOWING, expr_string) | (WITHIN, expr_string,
SURROUNDING, expr_string) | (WITHIN, PAST, expr_string) | (WITHIN, SAME,
DAY,
AS, expr_string) | (BEFORE, expr_string) | (AFTER, expr_string) | (EQUAL,
expr_string) | AT ) >

<!ELEMENT PAST EMPTY>

<!ELEMENT and EMPTY>

<!ELEMENT ENDIF EMPTY>

<!ELEMENT THEN EMPTY>

<!ELEMENT ELSE EMPTY>

<!ELEMENT WITHIN EMPTY>

<!ELEMENT unary_comp_op EMPTY>

<!ATTLIST unary_comp_op
        value

(PRESENT|NULL|BOOLEAN|NUMBER|TIME|DURATION|STRING|LIST)
#REQUIRED>

<!ELEMENT binary_comp_op (LESS_THAN | GREATER_THAN |
GREATER_THAN_OR_EQUAL |
LESS_THAN_OR_EQUAL | IN )>

```



```

<!ELEMENT of_func_op (of_read_func_op | of_noread_func_op )>

<!ELEMENT in_comp_op (IN)>

<!ELEMENT of_read_func_op EMPTY>

<!ATTLIST of_read_func_op
    value (AVERAGE|AVG|COUNT|EXIST|EXISTS|SUM|MEDIAN)
#REQUIRED>

<!ELEMENT of_noread_func_op EMPTY>

<!ATTLIST of_noread_func_op
    value (ANY|ALL|NO|SLOPE|STDDEV|VARIANCE|INCREASE|
PERCENT_INCREASE|DECREASE|PERCENT_DECREASE|INTERVAL|TIME|
ARCCOS | ARCSIN | ARCTAN   | COSINE   | COS | SINE       | SIN |
TANGENT   |
TAN | EXP | FLOOR | INT | ROUND | CEILING | TRUNCATE | LOG |
LOG10 |
ABS | SQRT | EXTRACT_YEAR | EXTRACT_MONTH | EXTRACT_DAY |
EXTRACT_HOUR |
EXTRACT_MINUTE | EXTRACT_SECOND | STRING |
EXTRACT_CHARACTERS | REVERSE )
#REQUIRED>

<!ELEMENT from_func_op EMPTY>

<!ATTLIST from_func_op
    value (NEAREST) #REQUIRED>

<!ELEMENT index_from_func_op EMPTY>

<!ATTLIST index_from_func_op
    value (INDEX_NEAREST) #REQUIRED>

```

```

<!ELEMENT from_of_func_op EMPTY>

<!ATTLIST from_of_func_op
    value ( MINIMUM|MIN|MAXIMUM | MAX | LAST | FIRST | EARLIEST
|
LATEST ) #REQUIRED>

<!ELEMENT index_from_of_func_op EMPTY>

<!ATTLIST indec_from_of_func_op
    value ( INDEX_MINIMUM | INDEX_MAXIMUM | INDEX_EARLIEST
| INDEX_LATEST )#REQUIRED>

<!ELEMENT as_func_op EMPTY>

<!ATTLIST as_func_op
    value ( NUMBER ) #REQUIRED>

<!ELEMENT boolean_value EMPTY>

<!ATTLIST boolean_value
    value ( true | false ) #REQUIRED>

<!ELEMENT time_value (NOW | iso_date_time | iso_date| EVENTTIME |
TRIGGERTIME)>

<!ELEMENT data_block (data_statement+)>

<!ELEMENT data_statement (empty | (data_assignment, event_assignment) )>

<!ELEMENT data_assignment ( ICD*, NDC*, LOINC*, string* )>

<!ELEMENT event_assignment ( empty | event* )>

<!ELEMENT LIONIC (#PCDATA)>

<!ATTLIST LIONIC
    CODE CDATA #REQUIRED
    var_name ID #REQUIRED

```

```

        timestamp CDATA #REQUIRED>
<!ELEMENT NDC (#PCDATA)>
<!ELEMENT ICD (#PCDATA)>
<!ATTLIST ICD
        CODE CDATA #REQUIRED
        var_name ID #REQUIRED
        timestamp CDATA #REQUIRED>
<!ELEMENT event ( (LOINC, update) | (LOINC, insert) | (ICD, found) | (NDC,
prescribed) | string)>
<!ATTLIST event
        timestamp CDATA #REQUIRED>
<!ELEMENT found EMPTY>
<!ELEMENT evoke_block (evoke_statement+)>
<!ELEMENT evoke_statement (empty | event_or | evoke_time |
qualified_evoke_cycle | CALL)>
<!ELEMENT event_or ( (event_or , or, event_any) | event_any)>
<!ELEMENT event_any ( identifier+ )>
<!ELEMENT evoke_time ( (evoke_duration, after,(time | time_of), event_any) |
(evoke_duration, after,(iso_date_time | iso_date)))>
<!ELEMENT qualified_evoke_cycle ( simple_evoke_cycle | (simple_evoke_cycle,
until, expr))>
<!ELEMENT simple_evoke_cycle ( every, evoke_duration, for, evoke_duration,
starting, evoke_time)>
<!ELEMENT evoke_duration (null | (number, duration_op))>
<!ELEMENT number ( (digits, exponent?) | (digits, point, exponent) | (digits,

```

```

point, digits, exponent) | (point, digits, exponent))>

<!ELEMENT exponent ( e, sign, digits)>

<!ELEMENT sign EMPTY>

<!ATTLIST sign
    value (POSITIVE|NEGATIVE) "POSITIVE">

<!ELEMENT digits (#PCDATA)>

<!ELEMENT digit EMPTY>

<!ATTLIST digit
    value (0|1|2|3|4|5|6|7|8|9) #REQUIRED>

<!ELEMENT duration_op EMPTY>

<!ATTLIST duration_op
    value
        (YEAR|YEARS|MONTH|MONTHS|WEEK|WEEKS|DAY|DAYS|HOUR|HOURS|M
        INUTE|MINUTES|SECOND|SECONDS) #REQUIRED>

<!ELEMENT identifier (#PCDATA)>

<!ATTLIST identifier
    value CDATA #REQUIRED
    ref CDATA #REQUIRED
    time CDATA #REQUIRED>

<!ELEMENT after EMPTY>

<!ELEMENT time EMPTY>

<!ELEMENT time_of EMPTY>

<!ELEMENT empty EMPTY>

<!ELEMENT null EMPTY>

<!ELEMENT CALL EMPTY>

```

<!ELEMENT until EMPTY>
 <!ELEMENT every EMPTY>
 <!ELEMENT for EMPTY>
 <!ELEMENT starting EMPTY>
 <!ELEMENT e EMPTY><!ELEMENT action_block (action_statement+)>
 <!ELEMENT action_statement (empty | (IF, action_if_then_else2) | (FOR,
 identifier, IN, expr, DO, action_block, ENDDO) | (WHILE, expr, DO, action_block,
 ENDDO) | call_phrase | (call_phrase, DELAY, expr) | (WRITE, expr) | (WRITE,
 expr, AT, identifier) | (RETURN, expr) | (email, institution, string) |
 (message))>
 <!ELEMENT action_if_then_else2 (expr, THEN, action_block, action_elseif)>
 <!ELEMENT action_elseif (ENDIF | (ELSE, action_block, ENDIF) | (ELSEIF,
 action_if_then_else2))>
 <!ELEMENT string (#PCDATA)>
 <!ELEMENT text (#PCDATA)>
 <!ELEMENT format_text (#PCDATA)>
 <!ELEMENT message (#PCDATA)>

 <!ELEMENT iso_date ((digit, digit, digit, digit, minus, digit, digit, minus,
 digit, digit)| string)>
 <!ELEMENT iso_date_time ((digit, digit, digit, digit, minus, digit, digit, minus,
 digit, digit, t, digit, digit, colon, digit, digit, colon, digit, digit,
 fractional_seconds, time_zone) | string)>
 <!ELEMENT t EMPTY>
 <!ATTLIST t

```

        value ( T | t ) #REQUIRED>

<!ELEMENT fractional_seconds (empty | (point, digit+))>

<!ELEMENT time_zone ( null | zulu | ( plus, digit, digit, colon, digit, digit)
| (minus, digit, digit, colon, digit, digit) )>

<!ELEMENT zulu EMPTY>

<!ATTLIST zulu
        value ( Z | z ) #REQUIRED>

<!ELEMENT term ( single_quote, string, single_quote)>

<!ELEMENT comments (#PCDATA)>

```