

Arden Syntax Implementation Guide

Release 2

**Contributors**

Robert A. Jenders, MD, MS; Charles Drew University & University of California, Los Angeles

Peter Haug, MD; Intermountain Healthcare & University of Utah

Karsten Fehre, MS; Medexter Healthcare

Klaus-Peter Adlassnig, PhD, MS; Medical University of Vienna & Medexter Healthcare

Tom Hooks, MT(ASCP), MBA, McKesson

**Project Sponsor**

HL7 Arden Syntax Work Group

**Co-Chairs**

Robert A. Jenders, MD, MS; Charles Drew University & University of California, Los Angeles

Peter Haug MD; Intermountain Healthcare & University of Utah

**HL7 Project #975**

**May 2014**

[1. Purpose 3](#_Toc383110382)

[2. Notes and disclaimer 5](#_Toc383110383)

[3. Arden Syntax: Context and History 6](#_Toc383110384)

[4. Syntax Description 8](#_Toc383110385)

[4.1. Fundamentals 8](#_Toc383110386)

[4.2. Language Concepts 9](#_Toc383110387)

[4.2.1. Data Types 10](#_Toc383110388)

[4.2.2. Statements 10](#_Toc383110389)

[4.2.3. Expressions 10](#_Toc383110390)

[4.2.4. Operators 11](#_Toc383110391)

[5. Basic Tasks (by Example) 12](#_Toc383110392)

[5.1. Sort a List of Objects 12](#_Toc383110393)

[5.2. Convert String to DateTime 13](#_Toc383110394)

[5.3. Calculate the Current Age in Years from a Given Birthday 14](#_Toc383110395)

[5.4. MLM-to-MLM Interaction 14](#_Toc383110396)

[6. Programming / Engineering Use Cases 16](#_Toc383110397)

[6.1. Guidelines 16](#_Toc383110398)

[7. System-Level Engineering Use Cases 20](#_Toc383110399)

[7.1. Standards-based stack for connecting Arden-based applications to an EHR 20](#_Toc383110400)

[7.2. Integrating CDS in PDMS with minimal effort 22](#_Toc383110401)

[7.3. Integrating CDS in a commercially available PDMS using proprietary interfaces 23](#_Toc383110402)

[8. Clinical Use Cases 24](#_Toc383110403)

[8.1. Drug-Disease Interaction 24](#_Toc383110404)

[8.2. Body Mass Index 26](#_Toc383110405)

[8.3. Abnormal Test Result Detection 27](#_Toc383110406)

[9. F.A.Q. 30](#_Toc383110407)

[10. References 37](#_Toc383110408)

# Purpose

The Arden Syntax for Medical Logic Systems is a structured, executable formalism for the explicit representation of the scientific, clinical, administrative and other knowledge used in clinical decision support systems. As such, it functions as a kind of programming language for such systems, allowing knowledge authors and clinical domain experts to implement knowledge-based interventions such as alerts and reminders, order sets, turnaround forms and the like in order to realize their quality improvement, clinical, administrative and public health objectives. Expressed in a way that resembles English-language syntax, the Arden Syntax also facilitates validation of knowledge bases by domain experts. In light of this utility, a number of vendors of clinical information systems and decision support systems have incorporated Arden Syntax into their products, leading to its adoption and use at numerous sites worldwide.

Despite the relative ease of use and functionality of the Arden Syntax, both novel and experienced users have questions regarding how best to use the Syntax to address particular clinical applications. Examples include generation of alerts related to drug-disease interactions, the implementation of multi-part clinical guidelines, immunization decision support and others. In addition, users and potential users of Arden Syntax may need to know how to use Arden to accomplish basic engineering tasks such as object manipulation, list sorting and the like.

The purpose of this implementation guide is to help answer these questions by providing, in addition to a summary of the Arden Syntax itself, ideas and examples regarding how Arden may be used in these different situations. This guide is not intended to be exhaustive in this regard, but it is meant to provide guidance on how to use the Arden Syntax to solve real-world challenges related to the implementation of clinical decision support. Further, while the summary of the Arden Syntax features presented herein provides the important highlights of this key standard, readers are directed to the actual Arden specifications for a complete definition of the language.

This implementation guide was composed when Arden Syntax v2.9 was the latest approved version of the standard and v2.10 was under development. While the examples and ideas featured here include elements of the Syntax that are new to these versions and may not be present in earlier versions, substantial parts of the implementation guide also leverage the backward compatibility of Arden, allowing users of earlier version also to make use of this implementation guide.

Finally, the reader should be aware that, while the authors have been diligent in providing useful, accurate content derived from real-world solutions already implemented in clinical decision support systems, no guarantee of accuracy or effectiveness is made regarding the examples and other information presented in this implementation guide. Any user or implementer of Arden Syntax assumes all liability regarding the use of any material contained in this guide.

The authors of this implementation guide hope that you find it a useful addition to other Health Level Seven publications related to clinical decision support in ways that allow you to make best use of the powerful and rich standard for representing clinical decision support knowledge that is the Arden Syntax.

# Notes and disclaimer

This implementation guide is not normative. Knowledge of the Arden Syntax standard is a prerequisite for reading this document. For a detailed description of this standard, we would like to refer to the Arden Syntax specification available on the Health Level Seven (HL7) International’s website.

# Arden Syntax: Context and History

Computer-based clinical decision support (CDS) has been shown to improve the quality of health care treatment and the performance of health care professionals. Clinical decision support involves delivering knowledge to decision-makers in clinical settings in order to improve the quality of decisions and the outcomes to which they lead. CDS sometimes is described in terms of the Five Rights: Delivering the right knowledge to the right person at the right time in the workflow in the right format via the right channel.

In order to provide computer-based CDS, the knowledge to be delivered must be represented in digital form. In this light, CDS can be divided into two broad classes: Services that facilitate delivery of knowledge and explicit, computable representations of the knowledge itself that can be shared via transfer and reuse. In the case of a knowledge delivery service, standards facilitate communication between electronic health record systems and other clinical software and knowledge sources, allowing connection of systems and sources from multiple vendors without having to negotiate and implement ad hoc methods for each connection. In the case of explicit knowledge encoding, standards facilitate sharing of knowledge by minimizing the changes necessary for the knowledge to be executed or used in different information systems.

The HL7 Infobutton standard is an example of a knowledge delivery service standard. It facilitates queries from users of electronic health record systems in the context of particular care activities and particular patients, providing knowledge from knowledge sources that is pertinent to these contexts. By contrast, examples of explicit knowledge encoding include the HL7 GELLO, Order Set and Arden Syntax standards.

A knowledge representation formalism constitutes one part of an overall CDS system. Units of knowledge encoded using the formalism are stored in the knowledge base (KB), independent of but linked to the inference engine or event monitor that executes units of the KB in combination with patient data to produce tailored, context-specific knowledge-based interventions that then can be delivered to the appropriate recipient such as a clinician, patient or administrator.

A prominent example of a knowledge formalism for encoding units of knowledge in the KB is the Arden Syntax for Medical Logic Systems. This is a computable language for encoding medical knowledge. It was previously adopted as a standard by the American Society for Testing and Materials (ASTM) as document E 1460, under subcommittee E31.15 Health Knowledge Representation. Adopted in 1992, it became Arden Syntax version 1.0.

Beginning in 1998, sponsorship of this standard was moved to HL7 International. Maintenance and further development of the standard is now overseen by the HL7’ s Arden Syntax Work Group. The Arden Syntax version 2.0 was formally adopted by HL7 and the American National Standards Institute (ANSI) in August 1999. Since then the standard has evolved, including the addition of new features and functionalities responding to the needs of users and vendors. Presently, the standard’s latest version–version 2.9–was adopted by HL7 and certified by ANSI in March 2013.

Arden Syntax uses medical logic modules (MLMs) as units of knowledge representation. Each of these MLMs contains sufficient knowledge to make a single medical decision. MLMs have been used to generate clinical reminders and alerts, interpretations, diagnoses and therapeutic advice, screening for clinical research, quality assurance functions, and administrative support. Using a computer program called an event monitor, MLMs run automatically, generating advice where and when it is needed.

This implementation guide describes the key features of the Arden Syntax and how it may be used in a variety of scenarios to deliver CDS.

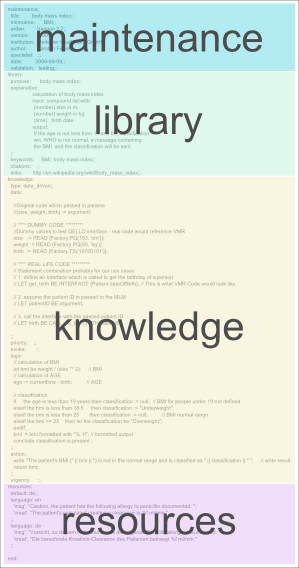
# Syntax Description

## Fundamentals

Medical knowledge in Arden Syntax is–as stated above–arranged within medical logic modules (MLMs), each of which contains sufficient knowledge to make a single decision. These MLMs are well organized and structured into categories and slots with specific content:

maintenance:  
 title: ;;  
 mlmname: ;;  
 arden: ;;  
 version: ;;  
 institution: ;;  
 author: ;;  
 specialist: ;;  
 date: ;;  
 validation: ;;  
library:  
 purpose: ;;  
 explanation: ;;  
 keywords: ;;  
 citations: ;;  
 links: ;;  
knowledge:  
 type: ;;  
 data: ;;  
 priority: ;;  
 evoke: ;;  
 logic: ;;  
 action: ;;  
 urgency: ;;  
resources:  
 default: ;;  
 language: ;;  
end:

The slots constituting an MLM are grouped into four categories: maintenance, library, knowledge, and resources. Each category starts with its name, followed directly by a colon (e.g., maintenance:). Both the four categories and the set of slots within each category have to appear in the correct order (see image above).



The maintenance category contains information unrelated to the MLM’s health knowledge and is used for MLM knowledge base maintenance and change control. The library category provides health personnel with explanatory information as well as links to relevant health literature related to the MLM’s health knowledge. The resources category specifies localized textual resources that can be used within the knowledge category. The knowledge category actually defines the MLM’s action, data access, and logic.

An MLM is identified by using the following three pieces of information: name, institution, version.

## Language Concepts

### Data Types

The basic function of an MLM is to retrieve patient data, manipulate the data, reach a decision, and possibly perform an action. The data necessary may come from various sources, for example via a direct query to the patient database, from a constant in the MLM, or resultant from an operation on other data. Available data types within the Arden Syntax are: Null, Boolean, Truth Value, Number, Time, Duration, String, List, Object and Fuzzy Sets. Every data item consists of a value part, a primary time part (e.g., time of data retrieval) and its applicability.

### Statements

Slots in Arden Syntax are structured, that is, composed of a set of statements. Each statement specifies a logical constraint or an action to be performed. In general, statements are carried out sequentially in the order that they are listed. All statements except for the last statement in a slot must end with a semicolon (;). Possible statements are:

* Read statement: reads data from external resources
* Event statement: assigns an institution-specific event definition to a variable
* Message statement: assigns an institution-specific message (e.g., an alert) to a variable
* Destination statement: assigns an institution-specific destination to a variable
* Interface statement: assigns an institution-specific foreign-function interface definition to a variable
* Assignment statement: places the value of an expression into a variable
* Write statement: sends a text or coded message to a destination
* Include statement: indicates that an external MLM may be consulted for object, MLM, event, interface variable, and resource definitions
* Conclude statement: ends execution in the logic slot
* Argument statement: accesses passed arguments
* Return statement: returns a result to the calling instance
* Loops: while- and for-loops
* If-then-else: permits conditional execution depending on the value of an expression
* Object statement: assigns object declaration to a variable
* Call statement (MLM, event, interface): permits an MLM to call other MLMs, events, or external interfaces
* Trigger: evokes a slot statement that defines how an MLM may be triggered

### 

### Expressions

Statements are composed of reserved words, special symbols, and expressions. Expressions may contain any of the following:

* Constant: data value that is explicitly represented
* Variable: a placeholder for a data value or special constructs (e.g. an event, MLM, message, or destination) and represents this value in any subsequent expressions. An assignment statement is used to assign a value to a variable
* Operator and arguments

Special case: curly brace expressions

As explained in the F.A.Q section of this document, institution-specific references to the local clinical data repository are embedded in so-called "curly braces" {}. This convention permits MLMs to be shared between institutions, since (ideally) only the codes/directives within the curly braces need to be revised – the logic in the remaining slots of the MLMs may remain unchanged. Here are some examples of the contents of curly braces from several implementations:

bmiEvent := EVENT {bmiEvent};

pathology\_upload := EVENT {'32506','32688'};

med\_order\_event := EVENT {A Med Order Entered:Antithrombotic Meds};

aminoglycoside\_order := read last  
 {'evoking','dam'="PDQORD1",'auxstr'="0013",  
 'constraints'="C\*\*\*\*",'status\_value'="A",  
 'display\_header'="R",'display\_comp'="V"; ; '23946'};

(ordName,ordFreq,ordStatus,ordPriority,cpFName,cpLName,

cpTitle,cpSeq,ordSeq)

:= READ last {An Active Order: Antithrombotic Therapy Exclusion};

last\_alert := read last (  
 {'dam'="PDQDEC1",'display\_header'="TX",'display\_comp'="";  
 'mlmself','mlm RF\_AND\_AMINOGLYCOSIDE'}  
 where it occurred within the past 3 months);

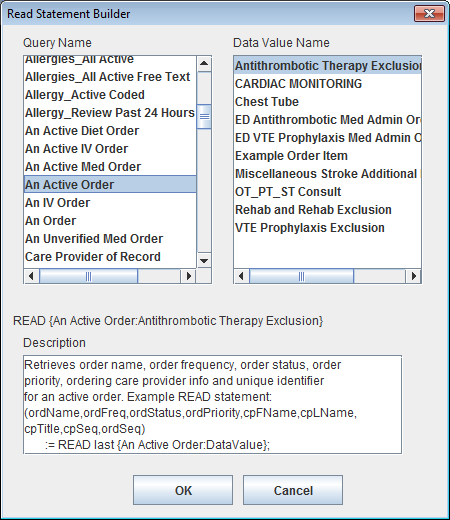
previous\_alert := READ last {Previous Alerts:Missing Antithrombotic Therapy for Stroke Patient}

where it occurred AFTER time\_between\_alerts ago;

email\_dest := destination {'email', 'name'="sidelir@cucis.cis.columbia.edu"};

carlene\_email := DESTINATION {Email:Carlene Dobber Email} ;

Some vendors have improved the setup and maintenance of curly braces by providing specific supporting utilities. For example, this "Statement Builder" permits users to create READ statements from a variety of options:



### Operators

Operators are used in expressions to manipulate data. They accept one or more arguments (data values) and they produce a result (a new data value). There are the following kinds of operators:

* List operators
* Logical operators
* Comparison operators
* String operators
* Arithmetic operators
* Temporal operators
* Aggregation operators
* Time operators
* Object operators

# Basic Tasks (by Example)

## Sort a List of Objects

maintenance:  
 title: sample\_sort\_objects;;  
 mlmname: sample\_sort\_objects;;  
 arden: Version 2.9;;  
 version: 1.0;;  
 institution: Medexter Healthcare;;  
 author: Karsten Fehre;;  
 specialist: ;;  
 date: 2013-11-13;;  
 validation: testing;;  
library:  
 purpose: ;;  
 explanation: ;;  
 keywords: ;;  
 citations: ;;  
 links: ;;  
knowledge:  
 type: data\_driven;;  
 data:   
 guidResult := object [ // result object: result of guidline query  
 compliance, // the compliance of the patient to a guidline  
 eOfIntervention, // the "ease of intervention"   
 text]; // the text  
 ;;  
 priority: ;;  
 evoke: ;;  
 logic:  
  
 result\_list := ();  
   
 resGuid1 := new guidResult with false, 12, "The patient is not in compliance with the guideline.";   
 result\_list := result\_list, resGuid1;  
   
 resGuid2 := new guidResult with true, 70, "The patient is in compliance with the guideline.";   
 result\_list := result\_list, resGuid2;  
   
 resGuid3 := new guidResult with true, 23, "The patient is in compliance with the guideline.";   
 result\_list := result\_list, resGuid3;  
   
 /\* sorting the result \*/  
 sorted\_list := ();  
 for en in result\_list do  
 inserted := false;  
 n := 1;  
 list\_size := count sorted\_list;  
 mylist := mylist, list\_size;  
 while not inserted do  
 if list\_size = 0 then  
 sorted\_list := sorted\_list, en;  
 inserted := true;  
 elseif n <= list\_size then  
 el := sorted\_list[n];  
 if ((el.compliance = false) and (en.compliance = false)) and (el.eOfIntervention > en.eOfIntervention) then  
 sorted\_list := sorted\_list[1 seqto (n-1)], en, sorted\_list[n seqto list\_size];  
 inserted := true;  
 elseif (el.compliance = true) and (en.compliance = false) then  
 sorted\_list := sorted\_list[1 seqto (n-1)], en, sorted\_list[n seqto list\_size];  
 inserted := true;  
 elseif ((el.compliance = true) and (en.compliance = true)) and (el.eOfIntervention > en.eOfIntervention) then  
 sorted\_list := sorted\_list[1 seqto (n-1)], en, sorted\_list[n seqto list\_size];  
 inserted := true;  
 endif;  
 else   
 sorted\_list := sorted\_list, en;  
 inserted := true;  
 endif;  
 n := n +1;  
 enddo;  
 enddo;  
 conclude true;   
 ;;  
 action:   
 return sorted\_list;  
 ;;  
 urgency: ;;  
end:

## Convert String to DateTime

In former versions of the Arden Syntax, the knowledge contained in the following MLM was necessary in order to allow the conversion of a string into a DateTime:

maintenance:  
 title: convert\_stringdate\_to\_datetime;  
 mlmname: convert\_stringdate\_to\_datetime;;  
 arden: Version 2.5;;  
 version: 1.0;;  
 institution: Medexter Healthcare;;  
 author: Karsten Fehre;;  
 specialist: ;;  
 date: 2013-11-13;;  
 validation: testing;;  
library:  
 purpose: ;;  
 explanation: ;;  
 keywords: ;;  
 citations: ;;  
 links: ;;  
knowledge:  
 type: data\_driven;;  
 data:   
 DOB\_str := argument;   
 ;;  
 priority: ;;  
 evoke: ;;  
 logic:  
 /\* for example DOB\_str := "1973-11-02T12:34:56"; \*/  
   
 year\_str := SUBSTRING 4 CHARACTERS FROM DOB\_str;   
 month\_str := SUBSTRING 2 CHARACTERS STARTING AT 6 FROM DOB\_str;   
 day\_str := SUBSTRING 2 CHARACTERS STARTING AT 9 FROM DOB\_str;   
 hour\_str := SUBSTRING 2 CHARACTERS STARTING AT 12 FROM DOB\_str;   
 minute\_str := SUBSTRING 2 CHARACTERS STARTING AT 15 FROM DOB\_str;   
 second\_str := SUBSTRING 2 CHARACTERS STARTING AT 18 FROM DOB\_str;  
   
 startDate := 1800-01-01T00:00:00;   
   
 DOB := startDate + (year\_str as number) years - 1800 years;   
 DOB := DOB + (month\_str as number) months - 1 month;   
 DOB := DOB + (day\_str as number) days - 1 day;   
 DOB := DOB + (hour\_str as number) hours;   
 DOB := DOB + (minute\_str as number) minutes;   
 DOB := DOB + (second\_str as number) seconds;  
   
 conclude true;   
 ;;  
 action:   
 return DOB;  
 ;;  
 urgency: ;;  
end:

As of the Arden Syntax version 2.8, this whole conversion can be simply done by using the following statement:

DOB := "1973-11-02T12:34:56" AS TIME

## Calculate the Current Age in Years from a Given Birthday

Given any date of birth, the corresponding age in years can be calculated using the following expression:

Age\_duration := currenttime - DOB;  
 Age\_yrs := Age\_duration/1 year;

## MLM-to-MLM Interaction

The following examples illustrate possible interactions between MLMs, such as importing definitions from and calling other MLMs.

This MLM solely contains an object definition:

maintenance:  
 title: sample\_object\_definition;;  
 mlmname: sample\_object\_definition;;  
 arden: Version 2.9;;  
 version: 1.0;;  
 institution: Medexter Healthcare;;  
 author: Karsten Fehre;;  
 specialist: ;;  
 date: 2013-11-13;;  
 validation: testing;;  
library:  
 purpose: ;;  
 explanation: ;;  
 keywords: ;;  
 citations: ;;  
 links: ;;  
knowledge:  
 type: data\_driven;;  
 data:   
 guidResult := object [ // result object: result of guidline query  
 compliance, // the compliance of the patient to this guidline  
 eOfIntervention, // the "ease of intervention"   
 text]; // the text  
 ;;  
 priority: ;;  
 evoke: ;;  
 logic: ;;  
 action:   
 ;;  
 urgency: ;;  
end:

By using the following two lines in the data slot of an MLM, it is possible to import and use the object definition of the MLM shown above within another MLM. This procedure allows for example large knowledge bases to share common definitions across all involved MLMs:

// MLM  
 mlmImport := mlm 'showcase\_spike\_definition' from institution "medexter";  
   
 // include  
 include mlmImport;

In large knowledge bases it is often necessary to be able to use particular calculations or parts of rules frequently. For this purpose, reusable parts of rules can be stored in separate MLMs that are accessible for all MLMs of the knowledge base. Additionally, this increases the readability and maintainability of the knowledge base and allows the reuse of outsourced parts in new knowledge bases.

Example for calling other MLMs

Here is a simple example of a called MLM. This may be called by a variety of other MLMs to standardize the output of patient information in email or SMS messages.

/\* Only call this MLM when you know that the calling rule is going to CONCLUDE TRUE

This rule should always be moved into either the Research or Testing folders.

In your CALLing rule . . .

Define the rule in the data slot:

get\_patient\_header := MLM 'Create Patient HIPAA Headers and Confid Msg' ;

Example call statement for rule:

(patient\_header, hipaa\_header, confidential\_msg) := CALL get\_patient\_header ;

\*/

MAINTENANCE:

title: Create Patient HIPAA Headers and Confid Msg ;;

mlmname: Create\_Patient\_HIPAA\_Headers\_and\_Confid\_Msg ;;

arden: Version 2;;

version: 1.00;;

institution: MPT ;;

author: MCK;;

specialist: ;;

date: 2006-01-15;;

validation: Testing;;

LIBRARY:

purpose: Use a single rule to create a standard patient header, a HIPAA header,

and a confidentiality message for notification messages. ;;

explanation:

This subroutine is called by a main rule to create a standard patient header.

The full header is to be used only internally at the site in order to be HIPAA compliant.

Otherwise, use the HIPAA Header. Can also include a Confidentiality Message as a footer in printer or email messages.

;;

keywords: Patient Header; called rule, subroutine;;

citations: ;;

links: ;;

KNOWLEDGE:

type: data-driven;;

data:

/\* All READs are done in this subroutine so it is independent of the Calling rule. \*/

(patTypeGroup, patService, patClass, patType) := READ last {Patient Type Information};

(patFacilityID, patDeptID, patRoomID, patBedID):= READ last {Patient Location};

(patSex, patLastName, patFirstName, patAcct, patMedRec, patMiddleName)

:= READ last {Person Info};

;;

evoke: // No trigger event - this MLM is 'triggered' by the rule that calls it.

;;

logic:

if (patMiddleName is not present) then

patient\_name := patLastName || ", " || patFirstName;

else

patient\_name := patLastName || ", " || patFirstName || " " patMiddleName;

endif;

patient\_header := NL || "Patient: " || patient\_name || NL

|| "Facility: " || patFacilityID || NL

|| "Unit: " || patDeptID || NL

|| "Location: " || patRoomID || NL

|| "Acct Number: " || patAcct || NL || NL ;

hipaa\_header := "RE: Patient in this Dept " || patDeptID || ", Room " || patRoomID ||

"; Acct # " || patAcct || NL ;

confidential\_msg := NL || NL || NL || "Confidentiality Notice: This message is for "

|| "the sole use of the intended recipient(s)and may contain confidential and "

|| "privileged information. Any unauthorized review, use, disclosure or distribution "

|| "is prohibited.";

CONCLUDE TRUE;

;;

action: /\* Returns a formatted patient header to the main rule in a single

variable as well as a hipaa header and confidential message \*/

RETURN (patient\_header, hipaa\_header, confidential\_msg) ;

;;

urgency: 49;;

END:

# Programming / Engineering Use Cases

## Guidelines

The following MLM is part of a larger knowledge base implementing a guideline for borreliosis detection and support. The calling instance sends a set of patient parameters to this MLM. The MLM checks, if relevant parameters for a decision are in the list. If at some point during the decision process a parameter necessary for the decision is missing, the MLM will create a so called 'question object' that is sent back to the calling instance. The calling instance can then use this question object to ask the corresponding medical staff for the specific parameter of the patient. With the additional parameter (additional to the original list) the MLM is called again. If no further parameters are needed for the final decision, the MLM will return a result or a recommendation based on the guideline.

maintenance:  
 title: borreliosis\_erythema\_migrans\_treatment;;  
 mlmname: borreliosis\_erythema\_migrans\_treatment;;  
 arden: Version 2.9;;  
 version: 1.08;;  
 institution: Medexter Healthcare GmbH, Vienna, Austria;;  
 author: Alexander Seitinger;;  
 specialist: Alexander Seitinger;;  
 date: 2008−08−20;;  
 validation: testing;;  
library:  
 purpose: borreliosis support;;  
 explanation:  
 managing all borreliosis EM treatment  
 ;;  
 keywords: borreliosis, decision support;;  
 citations: ;;  
 links: ;;  
knowledge:  
 type: data\_driven;;  
 data:  
 parameterlist := argument;  
 result\_obj := object[result, next\_questions];  
 question\_obj := object[q\_type, q\_label, q\_info];  
 ;;  
 priority: ;;  
 evoke: ;;  
 logic:  
 resultobj := new result\_obj;  
 resultobj.result := ();  
 resultobj.next\_questions := ();  
   
 age := −1;  
 gender := −1;  
   
 pregnant := −1;  
 breastfeeding := −1;  
 allergy\_amocicilline\_doxycycline := −1;  
 contraindication\_amocicilline\_doxycycline := −1;  
   
 for parameter in parameterlist do  
 if parameter.key eq "Age" then  
 age := parameter.val;  
 elseif parameter.key eq "Gender" then  
 gender := parameter.val;  
 elseif parameter.key eq "Pregnant" then  
 pregnant := parameter.val;  
 elseif parameter.key eq "Breastfeeding" then  
 breastfeeding := parameter.val;  
 elseif parameter.key eq "Allergy" then  
 allergy\_amocicilline\_doxycycline := parameter.val;  
 elseif parameter.key eq "Contraindication" then  
 contraindication\_amocicilline\_doxycycline := parameter.val;  
 endif;  
 enddo;

// if female: pregnant? breastfeeding?  
 if gender eq 1 then // female  
 if pregnant eq −1 then  
 question := new question\_obj;  
 question.q\_type := "polar\_bool";  
 question.q\_label := "Pregnant";  
 question.q\_info := "Pregnancy is the fertilization and development of one or more offspring.";  
   
 resultobj.result := resultobj.result, localized ’pregnant’;  
 resultobj.next\_questions := resultobj.next\_questions, question;  
 endif;  
 if breastfeeding eq −1 then  
 question := new question\_obj;  
 question.q\_type := "polar\_bool";  
 question.q\_label := "Breastfeeding";  
 question.q\_info := "Breastfeeding is the feeding of an infant or young child with breast milk directly from female human breasts.";  
   
 resultobj.result := resultobj.result, localized ’breastfeeding’;  
 resultobj.next\_questions := resultobj.next\_questions, question;  
 endif  
 endif;  
   
 // check for allergies  
 if allergy\_amocicilline\_doxycycline eq −1 then // then contraindication\_amocicilline\_doxycycline also -1  
 question := new question\_obj;  
 question.q\_type := "polar\_fuzzy";  
 question.q\_label := "Allergy";  
 question.q\_info := "mocicilline and/or Doxycycline allergy.";  
 question2 := new question\_obj;  
 question2.q\_type := "polar\_fuzzy";  
 question2.q\_label := "Contraindication";  
 question2.q\_info := "Amocicilline and/or Doxycycline contraindication.";  
 resultobj.result := resultobj.result, localized ’allerg\_amocicilline\_doxycycline’, localized ’contra\_amocicilline\_doxycycline’;  
 resultobj.next\_questions := resultobj.next\_questions, question, question2;  
 elseif allergy\_amocicilline\_doxycycline then  
 if pregnant then  
 resultobj.result := resultobj.result, localized ’erythema\_migrans\_alergy\_therapy\_pregnant’;  
 else  
 if age < 18 then // child  
 resultobj.result := resultobj.result, localized ’erythema\_migrans\_alergy\_therapy\_child’;  
 else  
 resultobj.result := resultobj.result, localized ’erythema\_migrans\_alergy\_therapy\_adult’;  
 endif;  
 endif;  
   
 question := new question\_obj;  
 question.q\_type := "polar\_bool";  
 question.q\_label := "Executed EM Therapy";  
 question.q\_info := "Patient took the medication for the adviced duration and dosage.";  
   
 resultobj.next\_questions := resultobj.next\_questions, question;  
 elseif contraindication\_amocicilline\_doxycycline then // same as allergy\_amocicilline\_doxycycline - should be solved with or  
 if pregnant then  
 resultobj.result := resultobj.result, localized ’erythema\_migrans\_alergy\_therapy\_pregnant’;  
 else  
 if age < 18 then // child  
 resultobj.result := resultobj.result, localized ’erythema\_migrans\_alergy\_therapy\_child’;  
 else  
 resultobj.result := resultobj.result, localized ’erythema\_migrans\_alergy\_therapy\_adult’;  
 endif;  
 endif;  
 question := new question\_obj;  
 question.q\_type := "polar\_bool";  
 question.q\_label := "Executed EM Therapy";  
 question.q\_info := "Patient took the medication for the adviced duration and dosage.";  
 resultobj.next\_questions := resultobj.next\_questions, question;  
 else // no allergy or contradiction  
 if pregnant then  
 resultobj.result := resultobj.result, localized ’erythema\_migrans\_therapy\_adult1’;  
 elseif breastfeeding then // same as pregnant  
 resultobj.result := resultobj.result, localized ’erythema\_migrans\_therapy\_adult1’;  
 else  
 if age < 18 then // child  
 resultobj.result := resultobj.result, localized ’erythema\_migrans\_therapy\_child1’;  
 if age > 9 then  
 resultobj.result := resultobj.result, localized ’erythema\_migrans\_therapy\_child2’;  
 endif;  
 else  
 resultobj.result := resultobj.result, localized ’erythema\_migrans\_therapy\_adult1’, localized ’erythema\_migrans\_therapy\_adult2’;  
 endif;  
 endif;  
 question := new question\_obj;  
 question.q\_type := "polar\_bool";  
 question.q\_label := "Executed EM Therapy";  
 question.q\_info := "Patient took the medication for the adviced duration and dosage.";  
 resultobj.next\_questions := resultobj.next\_questions, question;  
 endif;  
 conclude true;  
 ;;  
 action:  
 return resultobj;  
 ;;  
 urgency: ;;  
 resources:  
 default: en;;  
 language: en  
 ’pregnant’: "Check if patient is pregnant.";  
 ’breastfeeding’: "Ask patient if she is breastfeeding";  
 ’allerg\_amocicilline\_doxycycline’: "Check if patient has an Amocicilline or Doxycycline allergy.";  
 ’contra\_amocicilline\_doxycycline’: "Check if there are contraindications for Amocicilline or Doxycycline.";  
 ’erythema\_migrans\_alergy\_therapy\_pregnant’: "Choose Azithromycin or Erytromycin and take Ceftriaxone (under clinical surveillance) into consideration.";  
 ’erythema\_migrans\_alergy\_therapy\_adult’: "Choose Cefuroxim (2x500 mg p.o., 14−21 days), Azithromycin (1x500 mg p.o., 7−10 days) or Clarithromycin (2x500 mg p.o., 14−21 days)";  
 ’erythema\_migrans\_alergy\_therapy\_child’: "Choose Cefuroxim (30 mg/kg/day, 14−21 days), Azithromycin (5−10 mg/kg/day, 5−10 days) or Clarithromycin (2x500 mg p.o., 14−21 days)";  
 ’erythema\_migrans\_therapy\_adult1’: "Choose Amaxicillin (3x500 mg p.o., 14−21 days).";  
 ’erythema\_migrans\_therapy\_adult2’: "Doxycycline (2x100 mg p.o., 10 days) is also an option.";  
 ’erythema\_migrans\_therapy\_child1’: "For children use Amaxicillin (2−4 mg/kg/day, 14−21 days)";  
 ’erythema\_migrans\_therapy\_child2’: "For children older than 9 years Doxycycline (2−4 mg/kg/day, 10 days) is an option";;  
end:

# System-Level Engineering Use Cases

## 

## Standards-based stack for connecting Arden-based applications to an EHR

**Objective:**

Organizations may encounter the need to connect an Arden-based application to an arbitrary EHR. As a result, there may be interest in developing applications incorporating standards- based clinical logic representation and accessing EHR data in a schema-independent fashion. Note that “application” as used here may be either user-interactive (e.g., a browser-based application) or non-interactive (e.g., a background monitoring system).

**Solution: elements:**

* An authoring environment for development of computable knowledge in a shareable, standards-based representation (HL7 Arden Syntax)
* A free-standing, standards-based inferencing engine operating on standards-based representation of clinical logic (HL7 Arden Syntax) and using standards-based, schema-independent data-references (HL7 GELLO / vMR)
* A standards-based, schema-independent interface to the target EHR (HL7 GELLO / vMR) either through a mediation layer or a native implementation
* If a mediation layer is used, exposure of a data access service by the target EHR

**Target behavior summary for an interactive application:**

* The browser-based user interface uses standard web services (i.e., http/https) to connect to the middle tier CDS application (e.g., clinical reminder package or patient report card).
* The middle tier application invokes a CDS resource (inferencing service) to conduct the necessary clinical logic evaluation activity and then formats / produces results for return to the user. (Figure 1)
* The CDS resource in turn uses standard-web services to access a standards-based data layer to fetch data from the target EHR. (Figure 2)

**Sample implementation**

The approach described here was implemented as part of an innovation project carried out by the Veterans Health Administration using a GELLO/vMR layer developed by Medical Objects (Australia) and an Arden authoring and execution environment developed by MEDEXTER (Austria). Figures 1 and 2 reflect that project. The project successfully demonstrated a complete stack using HL7-compliant software and employing GELLO expressions to provide a solution to the “curly braces problem”.

**NOTE:** Mention of these commercial products does not represent endorsements of any particular company or product by VHA or HL7, and appear here only to be concrete about the nature of the architecture presented in this Engineering Use Case.

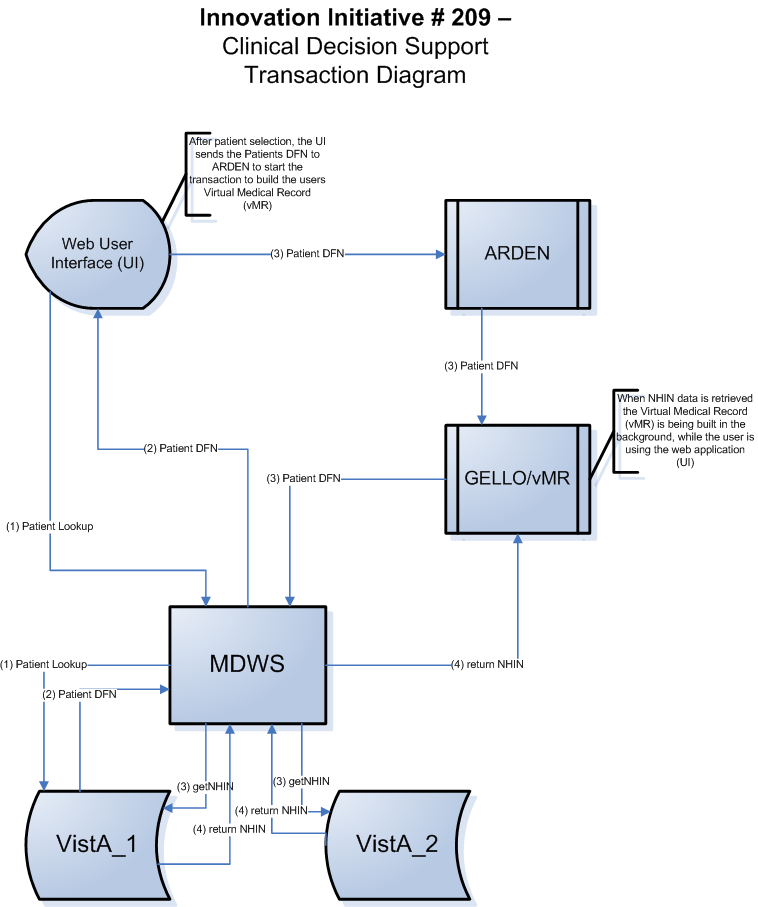


Figure 1: transaction diagram

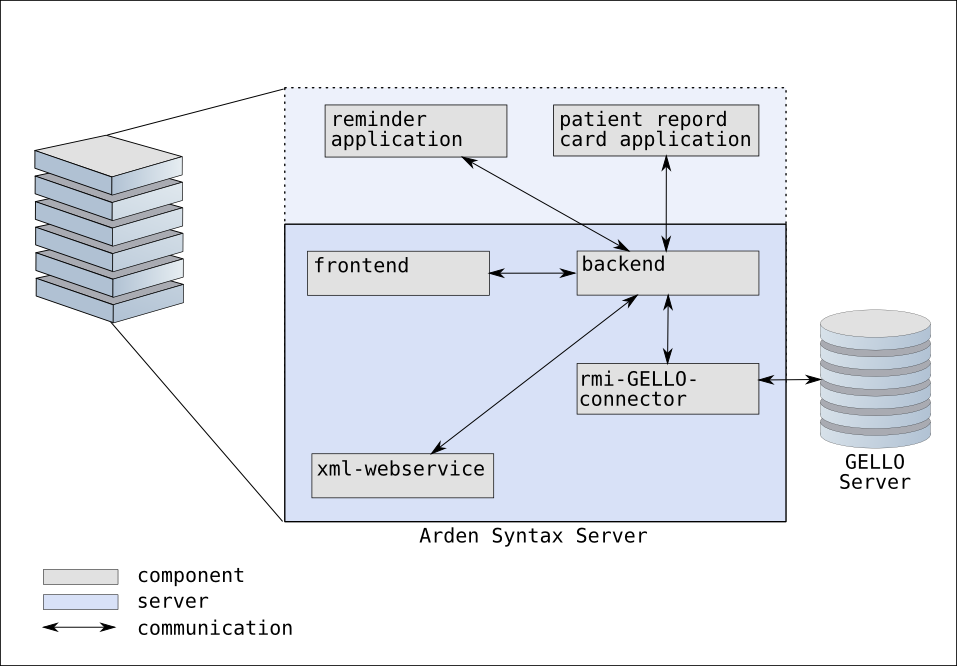


Figure 2: Arden Syntax server internal communication

## Integrating CDS in PDMS with minimal effort

**Objective:**

Integrate an Arden Syntax based CDS into an already available PDMS with minimal effort. Only specific topics should be covered by the CDS (e.g.: regular changing score calculations).

**Solution:**

* An authoring environment for development of computable knowledge in a shareable, standards-based representation (HL7 Arden Syntax)
* A free-standing, standards-based inferencing engine operating on standards-based representation of clinical logic (HL7 Arden Syntax)
* An interface to the inferencing engine that can be used from within the already available PDMS
* PDMS capability to use/consume external interfaces/services of the CDS

**Sample Implementation:**

An Arden Syntax engine equipped with an simple web-service interface that is able to receive the identification of an MLM to call and the needed data in simple XML format. Special forms (for each task, e.g.: score calculation, parameter check) in the PDMS are defined to collect the necessary data from the user or the internal storage and then sent to the Arden Syntax engine using the web-service interface. Returned results from the engine are displayed in the same form.

## Integrating CDS in a commercially available PDMS using proprietary interfaces

**Objective:**

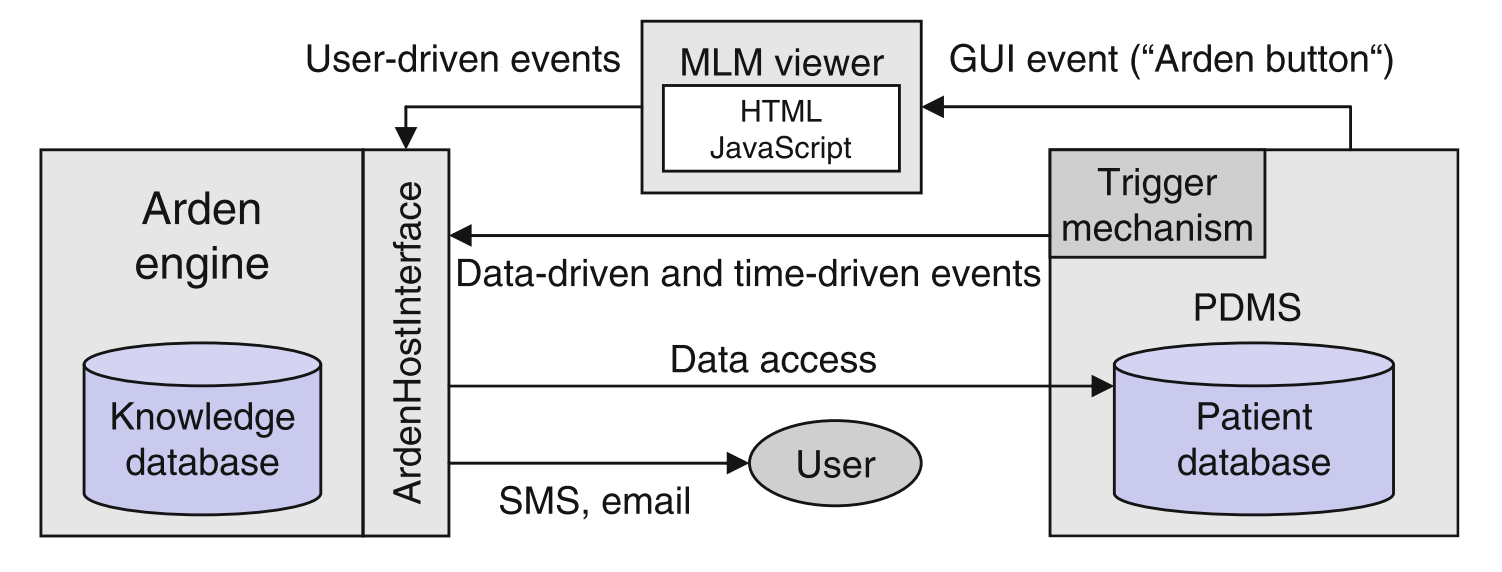
Integrate an Arden Syntax based CDS into an already available PDMS or provide an Arden Syntax based CDS access to data from an available PDMS.

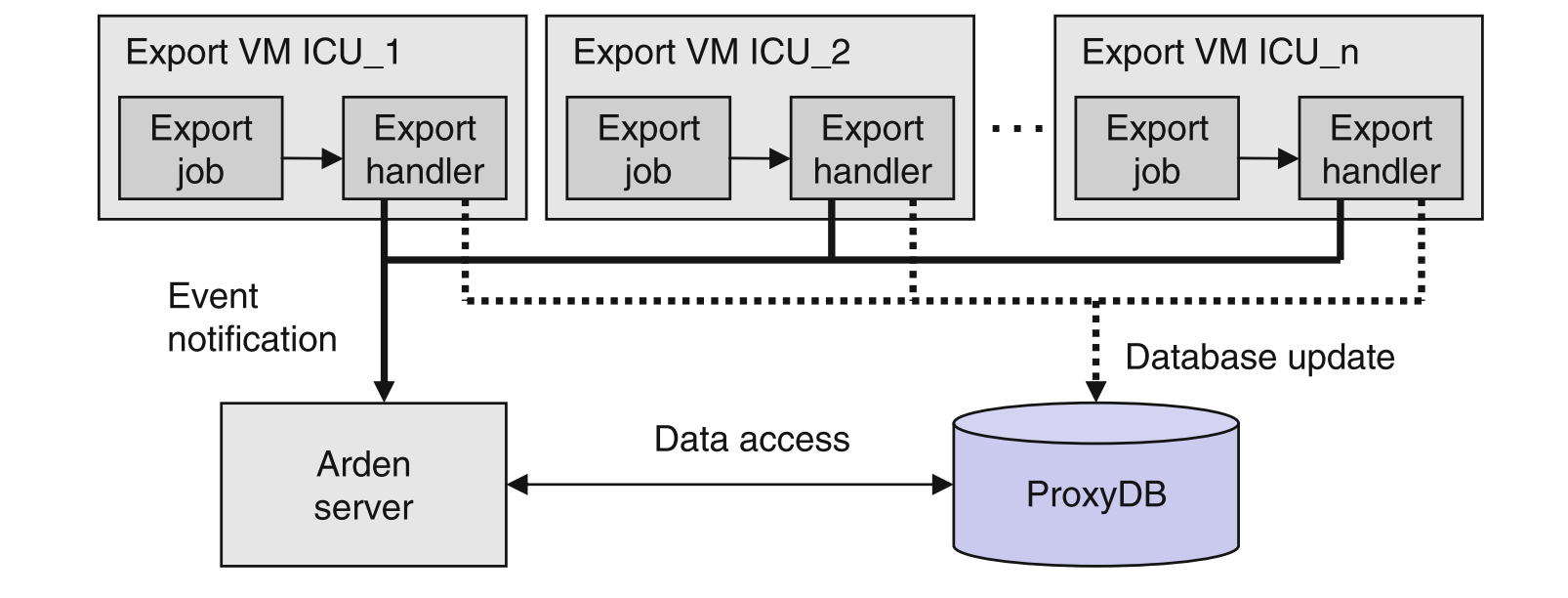
**Solution:**

* An authoring environment for development of computable knowledge in a shareable, standards-based representation (HL7 Arden Syntax)
* A free-standing, standards-based inferencing engine operating on standards-based representation of clinical logic (HL7 Arden Syntax)
* Implementation of an interface between the data source and the Arden Syntax execution engine
* An interface to the inferencing engine that can be used from within the already available PDMS
* PDMS capability to use/consume external interfaces/services of the CDS
* Export capability of the PDMS

**Sample Implementation:**

One implementation of this use case were done at the "University Hospital of Erlangen" (see [6] for more details). For this implementation the external interface of an Arden Syntax execution environment developed by Medexter Healthcare (Austria) to evaluate curly brace expressions were extended to access data exported from the PDMS.





Another implementation of this use case were done at the "University of Colorado Health". At this site external interfaces of the EPIC HIS were used to evaluate curly brace expressions. Curly brace expressions are sent through the specifically developed connector to the external interfaces of EPIC. Returning data is converted to execution internal data representation and then provided to the evaluating MLM.

# Clinical Use Cases

## Drug-Disease Interaction

Certain medications are contraindicated, must have their doses adjusted or must be monitored in the presence of certain diseases. The presence of at least some diseases may be revealed by laboratory testing. As laboratory test observations have been among the earliest discrete data captured by health information systems, many Arden Syntax MLMs have been written that provide alerts when a drug is ordered in the presence of certain conditions defined by laboratory test observations.

The following MLM demonstrates how to trigger an MLM based on a medication order and to then assess whether a disease (e.g., renal insufficiency) is present based on laboratory testing; alerting the user if this is so and if an alert about this has not been provided recently.

maintenance:  
  
 title: Screen for presence of renal insufficiency  
 and pharmacy order for an aminoglycoside antibiotic  
 (triggered by order for the aminoglycoside) ;;  
 filename: AMINOGLYCOSIDE\_AND\_RF;;  
 version: 1.05;;  
 institution: The Best Medical Center;;  
 author: James Best, MD;;  
 specialist: Francine Specialist, MD;;  
 date: 2013-02-20;;  
 validation: production;;  
  
library:  
  
 purpose: To determine if a patient has both laboratory evidence  
 of renal insufficiency (based on the serum creatinine) and an  
 active order for an aminoglycoside antibiotic;;  
  
 explanation: Aminoglycoside antibiotics, such as gentamicin,  
 tobramycin and amikacin, may cause or worsen renal  
 insufficiency. In addition, a typical dose of the antibiotic  
 must be adjusted when it is administered to a patient who  
 already has renal insufficiency. This module sends an alert if  
 one of this class of antibiotics 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. ;;  
  
 keywords: renal failure; aminoglycoside antibiotic;;  
 citations: ;;  
  
knowledge:  
  
 type: data-driven;;  
  
 data:  
 /\* evoke on storage of a pharmacy order \*/  
 storage\_of\_aminoglycoside\_order := event  
 {'30343','30345';'30343','30346'};  
  
 /\* read the aminoglycoside order that evoked the MLM \*/  
 aminoglycoside\_order := read last  
 {'evoking','dam'="PDQORD1",'auxstr'="0013",  
 'constraints'="C\*\*\*\*",'status\_value'="A",  
 'display\_header'="R",'display\_comp'="V"; ; '23946'};  
  
 /\* get the last appropriate creatinine \*/  
 raw\_creatinines := read last 3 from (  
 {'dam'="PDQRES2",'constraints'="C\*\*\*\*"; ; '32752'}  
 where they occurred within the past 3 months);  
  
 creatinine := last(raw\_creatinines where they are number);  
 creatinine\_boundary := 1.1; /\* upper reference range value \*/  
  
 /\* get the last related alerts \*/  
 last\_alert := read last (  
 {'dam'="PDQDEC1",'display\_header'="TX",'display\_comp'="";  
 'mlmself','mlm RF\_AND\_AMINOGLYCOSIDE'}  
 where it occurred within the past 3 months);   
;;  
  
 evoke:  
 /\* evoke on storage of a pharmacy order \*/  
 storage\_of\_aminoglycoside\_order;;  
  
 logic:  
  
 if (aminoglycoside\_order is null) or  
 (creatinine is null) then /\* insufficient data \*/  
 conclude false;  
 endif;  
  
 /\* avoid redundant alerts \*/  
 if last\_alert occurred after 2 weeks before time of  
 aminoglycoside\_order then  
 conclude false;  
 endif;   
  
 /\* check marker for renal insufficiency \*/  
 if creatinine <= creatinine\_boundary then  
 conclude false;  
 endif;  
  
 /\* otherwise creatinine is high and pt is on aminoglycoside \*/  
 conclude true;  
;;  
  
 action:  
  
 write "The patient has laboratory evidence of renal " ||  
 "insufficiency (serum creatinine of " || creatinine ||  
 " mg/dL on " || time of creatinine || "). Also, an active " ||  
 "order for an aminoglycoside antibiotic has been " ||  
 "recorded. Such antibiotics may cause or worsen renal " ||  
 "insufficiency, and special dosing may be required. " ||  
 "This may not be applicable to topical preparations. " ||  
 "Appropriate action should be taken as needed.";  
  
;;  
  
end:

## Body Mass Index

The following MLM allows calculation of the Body Mass Index (BMI):

maintenance:  
 title: body mass index;;  
 mlmname: BMI\_complex;;  
 arden: Version 2.5;;  
 version: 1.00;;  
 institution: Medexter Healthcare;;  
 author: Karsten Fehre;;  
 specialist: Harald Mandl;;  
 date: 2013-11-11;;  
 validation: testing;;  
library:  
 purpose: body mass index;;  
 explanation: calculation of body mass index  
 input: compound list with:  
 (number) size in m,  
 (number) weight in kg,  
 (time) birth date.  
 output: compound list with:  
 If the age is not less then 19 and the classification   
 wrt.WHO is not normal, a message containing   
 the BMI and the classification will be returned.   
 The classification follows the definition by the WHO, 2008.  
 The interpretation follows the BMI definition. For non adults  
 (age < 19) the definition by Kromeyer-Hauschild is used with the  
 3. and the 97. percentile.  
 ;;  
 keywords: BMI, body mass index;;  
 citations: ;;  
 links: <http://de.wikipedia.org/wiki/Body-Mass-Index>;;  
knowledge:  
 type: data\_driven;;  
 data:   
 (size, weight, birth) := argument; // input of MLM  
 bmiEvent := EVENT {bmiEvent};  
 ;;  
 priority: ;;  
 evoke:   
 bmiEvent;;  
 logic:  
   
 // calculation of BMI  
  
 let bmi be weight / (size \*\* 2); // BMI  
 age := currenttime - birth; // age  
   
 // classification wrt. WHO (only for adults)  
 if age < 19 years then classification := null;  
 elseif bmi < 16 then classification := localized 'strongunder';  
 elseif bmi < 17 then classification := localized 'modunder';  
 elseif bmi < 18,5 then classification := localized 'slightunder';  
 elseif bmi < 25 then classification := null;  
 elseif bmi < 30 then classification := localized 'obese';  
 elseif bmi < 35 then classification := localized 'obeseI';  
 elseif bmi < 40 then classification := localized 'obeseII';  
 else classification := localized 'obeseIII';  
 endif;  
  
 bmi := bmi formatted with localized 'msg'; // construct the localized message  
  
 conclude classification is present ; // if there is a classification, execute the action slot  
 ;;  
 action:   
 return bmi || classification || "."; // return result   
 ;;  
 urgency: ;;  
  
resources:  
 default: de;;  
 language: en  
 'msg' : "The patient's BMI %.1f is not in the normal range and is classified as ";  
 'strongunder': "severe thinness";  
 'modunder' : "moderate thinness";  
 'slightunder': "mild thinness";  
 'obese' : "pre-obese";  
 'obeseI' : "obese class I";  
 'obeseII' : "obese class II";  
 'obeseIII' : "obese class III"  
 ;;  
 language: de  
 'msg' : "Der BMI %.1f des Patienten ist nicht im normalen Bereich und wird klassifiziert als ";  
 'strongunder': "starkes Untergewicht";  
 'modunder' : "mäßiges Untergewicht";  
 'slightunder': "leichtes Untergewicht";  
 'obese' : "Präadipositas (Übergewicht)";  
 'obeseI' : "Adipositas Grad I";  
 'obeseII' : "Adipositas Grad II";  
 'obeseIII' : "Adipositas Grad III"  
 ;;  
end:

## Abnormal Test Result Detection

MLMs can be used to alert clinicians to potentially serious test results that may require follow-up. These may be numeric laboratory observations or they may be narrative or coded results associated with a report. The following MLM demonstrates how diagnostic codes associated with the report of the analysis of a Papanicolaou smear can be detected and an alert sent.

maintenance:

title: Abnormal PAP smears and cervical biopsies.;;

filename: pap\_ monitoring;;

version: 1.02;;

institution: Best Medical Center;;

author: Robert Jones, M.D.;;

specialist: Michael Smith, M.D;;

date: 2014-01-27;;

validation: production;;

library:

purpose: Screen/follow up patients with abnormal PAP smears and cervical biopsies;;

explanation: Patients who have atypical cells, suspicious cells or malignant cells present in their PAP smear need a follow up PAP smear or cervical biopsy within 3 months. Patients who have abnormal findings on cervical biopsies need follow up within 3 months.;;

keywords: Papanicolau smear, cervical biopsy, cytology ;;

citations: ;;

knowledge:

type: data-driven;;

data:

/\* evoking record = all pathology reports \*/

pathology\_upload := EVENT {'32506','32688'};

/\* patient name \*/

name := read last

{'dam'="GYDAPMP"; "HPBASIC"; "HNAME"};

/\* patient mrn \*/

mrn := read last

{'pcodes'="null mrn "};

/\* patient address - phone number \*/

(address, state, city, zip, phone) := read last

{'dam'="GYDAPMP"; "HADDRESS"; "HADDADDR"; "HSTATE"; "HADDCITY";

"HZIP"; "HPHONNO" };

/\* get diagnoses from evoking report \*/

path\_codes := READ

{'evoking','dam'="GYDATXP"; "PATH"; "PTECH/CLERK"};

/\* get details re: evoking report \*/

path\_accno := read last

{'pcodes'="now event\_db\_key "};

/\* email for research log \*/

email\_dest := destination

{'email', 'name'="sidelir@cucis.cis.columbia.edu"};

;;

evoke: pathology\_upload;;

logic:

if any(path\_codes = "AG9,1")

then

Specimen := "Papanicolau Smear";

Path\_DX := "Atypical cells present";

conclude true;

elseif any(path\_codes = "SG9,1")

then

Specimen := "Papanicolau Smear";

Path\_DX := "Suspicious cells present";

conclude true;

elseif any(path\_codes = "PSG9,1")

then

Specimen := "Papanicolau Smear";

Path\_DX := "Positive for malignant cells";

conclude true;

elseif any(path\_codes = "C19,1")

then

Specimen := "Papanicolau Smear";

Path\_DX := "CIN 1";

conclude true;

elseif any(path\_codes = "C29,1")

then

Specimen := "Papanicolau Smear";

Path\_DX := "CIN 2";

conclude true;

elseif any(path\_codes = "C39,1")

then

Specimen := "Papanicolau Smear";

Path\_DX := "CIN 3";

conclude true;

else;

endif;

;;

action:

write "\n MRN: " || mrn ||

"\n Patient Name: " || name ||

"\n Address: " || address ||

"\n " || city || ", " || state || " " || zip ||

"\n Phone: " || phone ||

"\n" ||

"\n Path Acc #: " || path\_accno ||

"\n Date: " || last(time of path\_codes) ||

"\n Specimen: " || Specimen ||

"\n Path\_DX: " || Path\_DX

at email\_dest;

;;

urgency: 50;;

end:

8.4 Detection of possible patient deterioration

The ability of MLMs to quickly identify abnormal test results (Section 8.3 Abnormal Test Result Detection) can be extended to detect patterns of results and vital signs to give an "early warning" that a patient's condition is deteriorating. This "MEWS Alert" evaluates eight different clinical parameters, calculates an aggregate MEWS score, and if elevated, immediately notifies the clinician and/or "Rapid Response" team.

MAINTENANCE:

title: MEWS Alert;;

mlmname: MEWS\_Alert;;

arden: Version 2;;

version: 1.40;;

institution: InSight;;

author: Tom Hooks, McKesson

Mike Jones, University of Colorado

Michelle Kearney, John Muir Health

Colleen Nowlin, Elkhart General

Rob O'Daniel, Ector County;;

specialist: ;;

date: 2011-08-04;;

validation: Testing;;

LIBRARY:

purpose: This rule checks vital signs and identifies patients at imminent risk for deterioration.;;

explanation: This rule was created by a group of InSight HCA SIG members based on Early Warning Score literature (MEWS and ViEWS -- see references below. The rule detects abnormal vital signs, calculates a weighted aggregate "MEWS Score" and notifies caregivers when a patient is at risk of deterioration. The appropriate criteria to use (mews\_threshold) is hospital-specific, and is based on the number of vital signs being checked - typical threshold values are 4 to 6. Refer to the articles below, or Google 'Modified Early Warning System' (MEWS) for more information..

Original MEWS criteria (see citation 1)

3 2 1 0 1 2 3

Pulse rate (bpm) < 40 40-50 51-100 101-110 111-129 > 129

Respiratory rate < 9 9-14 15-20 21-29 >= 30

Temperature C < 35.1 35.1-36 36.1—38 38.1-38.5 >38.5

Systolic BP < 70 71-80 81-100 101-199 >= 200

Neuro score (AVPU) Alert voice pain Unresponsive

Additional ViEWS criteria (see citation 2):

O2 Saturation <=91 91.1-93 93.1-95 >95

Supplemental O2 Room Air Any O2

;;

keywords: MEWS ;;

citations: 1. Subbe CP, Davies RG, Williams E et al: Effect of introducing the Modified Early Warning score on clinical outcomes, cardio-pulmonary arrest and intensive care utilization in acute medical admissions. Anaesthesia. 2003; 58:775-803.

2. Prythercha DR, Smith GB, Paul E. Schmidt PE, Peter I. Featherstone PI: ViEWS -- Towards a national early warning score for detecting adult inpatient deterioration. Resuscitation. 2010; 81:932-937.

;;

links: ;;

KNOWLEDGE:

type: data\_driven;;

data: // Enter the appropriate Event here

triggering\_event := EVENT {A Charting Result Entered:Pulse};

// Read mappings for results

(HeartRateText):= READ last {A Text Charting Result:Pulse}

where it occurred after result\_interval AGO;

(RespRateText):= READ last {A Text Charting Result:Respirations}

where it occurred after result\_interval AGO;

(TemperatureText):= READ last {A Text Charting Result:Temperature}

where it occurred after result\_interval AGO;

(BPText):= READ last {A Text Charting Result:Blood Pressure}

where it occurred after result\_interval AGO;

(O2SatText):= READ last {A Text Charting Result:O2Sat}

where it occurred after result\_interval AGO;

// Indicates that patient is receiving O2 Therapy. Use EITHER a charted result or an active order (not both)

(O2Admin):= READ last {A Text Charting Result: O2Admin}

where it occurred after result\_interval AGO;

// (O2Admin):= READ last {An Active Order:Supplemental O2};

(LOCText):= READ last {A Text Charting Result:Level of Conciousness}

where it occurred after result\_interval AGO;

/\*Read mappings for patient data.\*/

(patSex, patLastName, patFirstName, patAcct, patMedRec) := READ last {Person Info};

(patFacilityID, patDeptID, patRoomID,patBedID) := READ last {Patient Location};

(patTypeGroup, patService, patClass, patType) := READ last {Patient Type Information};

// Previous notifications

previous\_notification:= READ last{Previous Notifications:MEWS Alert}

where it occurred after time\_between\_alerts AGO;

//Destinatiions

admin\_email := DESTINATION{Email:Example Email Address};

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Revise these variables as needed

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Designate the specific patient type codes and depts to INCLUDE when processing. If //null, all types/depts will be included

include\_patType :=( "I/P", "IP", "OBS" , "OIB") ;

include\_depts := ("V5W","V6W","V6E","V7E","V7W","V8E","V8W","V9E","V9W");

// This is the timeframe used for recent results. Any result older than this will

// be disregarded

result\_interval := 4 hours;

// set to null to disable check for previous notifications

time\_between\_alerts := 4 hours;

// The results you want to check must be set to true

check\_pulse := true;

check\_respiration := true;

check\_temp := true;

check\_SysBP := true;

check\_map := false; // Mean Arterial Pressure

check\_LOC := false; // Level of concsiousness (AVPU score)

check\_O2sat := true;

check\_O2\_admin := true;

// MEWS score threshold. A score equal to or greater than this will generate an alert.

// This is normally 4 to 6. Set it to zero to force the rule to ALWAYS conclude true.

mews\_threshold := 5;

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Initialized variables -- You should NOT have to change any of these

mews\_score := 0;

msg := "";

detail\_msg := "";

mews\_msg := "";

na := "N/A" formatted with "%12s "; // displays if result is not available

NL := "

";

;;

evoke: triggering\_event;

;;

logic:

// Add your normal validation checks here. These are examples

// Check for previous notifications.

IF time\_between\_alerts is present THEN

IF previous\_notification is present THEN

CONCLUDE FALSE;

ENDIF;

ENDIF;

// Check the patient type against the INCLUDE\_patType list (if defined).

IF include\_patType is present THEN

IF patType is NOT in include\_patType THEN

CONCLUDE FALSE;

ENDIF;

ENDIF;

// Check the patient's department against the INCLUDE\_depts\_list list (if defined).

IF include\_depts is present THEN

IF patDeptID is NOT in include\_depts THEN

CONCLUDE FALSE;

ENDIF;

ENDIF;

// End of validation. Start checking results now

IF check\_pulse THEN

// PULES/HEARTRATE PLUGIN

// Variables with a "\_f " suffix are formatted for output.

result\_name := "Pulse: ";

result\_name\_f := result\_name formatted with "%-22s";

result\_score := 0;

abnormal\_flag := " ";

IF HeartRateText is present THEN

result\_text := HeartRateText;

result\_time := time of HeartRateText;

heartRate:= HeartRateText AS NUMBER;

IF heartRate < 40 THEN

result\_score := 2 ;

ELSEIF heartRate <= 50 THEN

result\_score := 1 ;

ELSEIF heartRate <= 100 THEN

result\_score := 0 ;

ELSEIF heartRate <= 110 THEN

result\_score := 1 ;

ELSEIF heartRate <= 129 THEN

result\_score := 2 ;

ELSEIF heartRate > 129 THEN

result\_score := 3 ;

ENDIF ;

IF result\_score > 0 THEN

mews\_score := mews\_score + result\_score;

abnormal\_total := abnormal\_total + 1;

abnormal\_flag := "\*";

ENDIF;

// Build the default, detail, and MEWS messaqes

msg := msg || result\_name || result\_text ||", ";

result\_text\_f := result\_text formatted with "%12s ";

detail\_msg := detail\_msg || result\_name\_f || result\_text\_f || abnormal\_flag

|| " " || result\_time || NL;

mews\_integer\_f := result\_score formatted with "%5.f";

mews\_msg := mews\_msg || result\_name\_f || result\_text\_f || abnormal\_flag

|| " " || result\_time || mews\_integer\_f || NL;

ELSE

// The result is not present -- just add N/A to the messages

msg := msg || result\_name || "N/A, ";

detail\_msg := detail\_msg || result\_name\_f || na || NL;

mews\_msg := mews\_msg || result\_name\_f || na || NL;

ENDIF;

// End of Pules/HeartRate plugin

ENDIF;

IF check\_respiration THEN

// RESP RATE PLUGIN

// Variables with a "\_f " suffix are formatted for output.

result\_name := "Respiration: ";

result\_name\_f := result\_name formatted with "%-22s";

result\_score := 0;

abnormal\_flag := " ";

IF respRateText is present THEN

result\_text := respRateText;

result\_time := time of respRateText;

respRate:= respRateText AS NUMBER;

// Calculate MEWS score

if respRate >= 30 then

result\_score := 3;

elseif respRate >= 21 then

result\_score := 2;

elseif respRate >= 15 then

result\_score := 1;

elseif respRate >= 9 then

result\_score := 0;

else

result\_score := 2;

endif;

if result\_score > 0 then

mews\_score := mews\_score + result\_score;

abnormal\_total := abnormal\_total + 1;

abnormal\_flag := "\*";

endif;

// Build the default, detail, and MEWS messaqes

msg := msg || result\_name || result\_text ||", ";

result\_text\_f := result\_text formatted with "%12s ";

detail\_msg := detail\_msg || result\_name\_f || result\_text\_f || abnormal\_flag

|| " " || result\_time || NL;

mews\_integer\_f := result\_score formatted with "%5.f";

mews\_msg := mews\_msg || result\_name\_f || result\_text\_f || abnormal\_flag

|| " " || result\_time || mews\_integer\_f || NL;

ELSE

// The result is not present -- just add N/A to the messages

msg := msg || result\_name || "N/A, ";

detail\_msg := detail\_msg || result\_name\_f || na || NL;

mews\_msg := mews\_msg || result\_name\_f || na || NL;

ENDIF;

// End of Resp Rate plugin

ENDIF;

Additional "plugins" for Temperature, Systolic BP, MAP, Level of Conciousness, Oxygen Saturation, and O2 Administration have been omitted here for brevity

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// All results have been evaluated. Here is the standard end of the rule

IF mews\_score >= mews\_threshold THEN

// Replace this with your normal patient header logic

patient\_header := "Patient: " || patLastName || ", " || patFirstName || NL

|| "Facility: " || patFacilityID || NL

|| "Location: " || patRoomID || NL

|| "Account Number: " || patAcct || NL || NL ;

mews\_score\_f := mews\_score formatted with "%2.f";

msg\_summary := "This patient has a MEWS score of " || mews\_score\_f || " based on these results: " || NL;

mews\_report\_header := " MEWS" || NL ||

"Result Value Chart Time Score"|| NL ||

"--------------- -------- -------------- -----"|| NL;

mews\_report\_footer := " -----"|| NL || " Total "|| mews\_score\_f;

msg := msg\_summary || msg;

detailed\_mews\_msg := patient\_header || msg\_summary || NL || mews\_report\_header ||

mews\_msg || mews\_report\_footer;

CONCLUDE TRUE;

ENDIF;

;;

action:

WRITE msg;

WRITE detailed\_mews\_msg || "^MEWS Alert" AT admin\_email;

;;

urgency: 75;;

END:

# F.A.Q.

**Q: What is the Arden Syntax standard?**

A: The Arden Syntax standard allows the sharing of computerized health knowledge bases among personnel, information systems, and institutions. Its scope has been limited to those knowledge bases that can be represented as a set of discrete modules. Each module, referred to as a Medical Logic Module (MLM), contains sufficient knowledge to make a single medical decision. Contraindication alerts, management suggestions, data interpretations, treatment protocols, and diagnosis scores are examples of health knowledge representable using MLMs.

Additionally, each MLM contains management information to help maintain a knowledge base consisting of MLMs as well as links to other knowledge sources. With this format, MLMs can be created directly by health personnel and can immediately be used by information systems conforming to this specification.

History: The Arden Syntax evolved from alerts and reminder systems at the LDS Hospital in Salt Lake City (i.e., the HELP system), the Regenstrief Institute in Indianapolis (the CARE system), the Columbia University Medical Center in New York (the first Arden Syntax system), and several other academic efforts. The group first met at the Arden Homestead in Harriman, NY, hence the name. The Arden Syntax was born out of the realization that the power of those alerts and reminder systems lies in the knowledge, and that there was a need to make this knowledge portable, shareable, between clinical information systems. Many commercial vendors since adopted the standard and included it in their products (see below).

The first version of the Arden Syntax was administered and issued by the American Society for Testing and Materials (ASTM, see below). Since 1998, the Arden Syntax group is part of the Health Level Seven International (HL7) organization, home of many widely accepted standards in health care informatics. The Arden Syntax Work Group administers the Arden Syntax standard.

**Q: How do I get a copy of the Arden Syntax standard?**

A: Visit the [Arden Syntax product page](http://www.hl7.org/implement/standards/product_brief.cfm?product_id=290) at the HL7 International website.

**Q: How do I join the Arden Syntax Work Group?**

A: The Arden Syntax Work Group is part of the HL7 International organization. For information on HL7 meetings, contact HL7 at:

HL7 International

3300 Washtenaw Avenue

Suite 227

Ann Arbor, MI 48104-4250

phone: (734) 677-7777

fax: (734) 677-6622

email: hq@hl7.org

Web site: [hl7.org](http://www.hl7.org/)

<http://www.hl7.org/>

To join the HL7 Arden Syntax Work Group (ardensyntax@lists.hl7.org) or Clinical Decision Support Work Group (dss@lists.hl7.org) listservs, see instructions on the HL7 website.

Contact persons for information on Arden Syntax are the two Arden Syntax Work Group co-chairs:

Robert Jenders, MD, MS, FACP, FACMI

Center for Biomedical Informatics and Department of Medicine, Charles Drew University

Department of Medicine, University of California, Los Angeles

voice: (310) 761-4700

email: jenders@ucla.edu

and

Peter Haug, MD

University of Utah and Intermountain Healthcare

email: peter.haug@imail.org

**Q: Which vendors of clinical information systems use the Arden Syntax in their clinical decision support applications?**

A: The following vendors have applications available that support the Arden Syntax (as of July 2013):

Allscripts [[1]](http://www.allscripts.com/)

McKesson [[2]](http://www.mckesson.com/)

Medexter Healthcare [[3]](http://www.medexter.com/)

Siemens Healthcare [[4]](http://www.medical.siemens.com/)

**Q: Which health care organizations have Arden-Syntax-based systems in use?**

A: The list below includes the main installed sites per vendor (in alphabetical order) and is not exhaustive. Please contact the vendor directly for any additional information. This list is dated December 2013.

* Alamance Regional Medical Center, Burlington, NC (Allscripts)
* Sarasota Memorial Hospital, Sarasota FL (Allscripts)
* Covenant Health, Knoxville, TN (McKesson)
* JFK Medical Center, Edison, NJ (McKesson)
* Mississippi Baptist Health Systems, Jackson, MS (McKesson)
* St. Mary's Hospital, Waterbury, CT (McKesson)
* St. Mary's Medical Center, Knoxville, TN (McKesson)
* St. Vincent's Hospital, Birmingham, AL (McKesson)
* Vienna General Hospital/Medical University of Vienna, Austria (Medexter Healthcare)
* Universitätsklinikum Erlangen, Germany, (Medexter Healthcare)
* University of Colorado Hospital Authority, CO (Medexter Healthcare)
* Chester County Hospital - West Chester, PA (Siemens Healthcare)
* McLeod Regional Medical Center, Florence, SC (Siemens Healthcare)
* Ohio State University, Columbus OH (Siemens Healthcare)

**Q: Where can I get MLMs for my Arden Syntax decision support system?**

A: Your software or systems vendor may also supply you with a set of MLMs, if Arden Syntax MLMs are part of the company’s applications.

**Q: What implementation tools are available to help me use the Arden Syntax in my systems?**

A: Most commercial applications are developed by vendors for use primarily within their own environment. As of December 2013, only the Medexter Healthcare company offers Arden Syntax software systems and/or components that allow you to embed them into your own clinical systems or environments, and can even be embedded into other vendor’s clinical systems.

**Q: What are the benefits of Arden-Syntax-based clinical decision support systems?**

A: Clinical alerts and reminder systems have proven to be effective in improving the quality and reducing the cost of health care. Many publications demonstrate the effect of these systems in areas such as effective medication use, reduction of adverse drug events, infection control, preventive care and wellness, etc. See below for a list of publications documenting the effect of clinical decision support on quality and cost of health care.

Through the use of Arden Syntax, the knowledge that drives these systems becomes portable. As a result, hospitals and health care providers can incorporate knowledge into their systems that was developed and refined elsewhere, without having to “reinvent the wheel”.

**Selected bibliography on clinical decision support:**

* Bates DW, Cohen M, Leape LL, Overhage JM, Shabot MM, Sheridan T. Reducing the frequency of errors in medicine using information technology. J Am Med Inform Assoc 2001;8(4):299-308.
* Bates DW, Gawande AA. Improving safety with information technology. N Engl J Med 2003;348(25):2526-2534.
* Evans RS, Pestotnik SL, Classen DC et al. A computer-assisted management program for antibiotics and other antiinfective agents. N Engl J Med1998;338(4):232-238.
* Johnston ME, Langton KB, Haynes RB, Mathieu A. Effects of computer-based clinical decision support systems on clinician performance and patient outcome. A critical appraisal of research. Ann Intern Med 1994;120:135-142.
* McDonald CJ, Overhage JM, Tierney WM, Abernathy GR, Dexter PR. The promise of computerized feedback systems for diabetes care. Ann Intern Med 1996;124(1 pt 2):170-174.
* Pestonik SL, Classen DC, Evans RS, Burke JP. Implementing antibiotic practice guidelines through computer-assisted decision support: clinical and financial outcomes. Ann Intern Med 1996;124:884-890.
* Rind DM, Safran C, Phillips RS, Wang Q, Calkins DR, Delbanco TL, Bleich HL, Slack WV. Effect of computer-based alerts on the treatment and outcomes of hospitalized patients. Arch Intern Med 1994;154:1511-1517.
* Shea S, DuMouchel W, Bahamonde L. A meta-analysis of 16 randomized controlled trials to evaluate computer-based clinical reminder systems for preventive care in the ambulatory setting. J Am Med Inform Assoc 1996;3(6):399-409.
* Shea S, Sideli RV, DuMouchel W, Pulver G, Arons RR, Clayton PD. Computer-generated informational messages directed to physicians: effect on length of hospital stay. J Am Med Informatics Assoc 1995;2:58-64.
* Somkin CP, Hiatt RA, Hurley LB, Gruskin E, Ackerson L, Larson P. The effect of patient and provider reminders on mammography and Papanicolaou smear screening in a large health maintenance organization. Arch Intern Med 1997;157:1658-1664.
* Weingarten SR, Riedinger MS, Conner L, Lee TH, Hoffman I, Johnson B, Ellrodt AG. Practice guidelines and reminders to reduce duration of hospital stay for patients with chest pain. An interventional trial. Ann Intern Med 1994;120:257-263.

**Q: Where can I read more about the Arden Syntax?**

A: A number of research papers have been published involving the Arden Syntax or systems that use it. (see also list of references, chapter 9)

**Selected bibliography on Arden Syntax:**

* Hripcsak G. Writing Arden Syntax medical logic modules. Computers in Biology and Medicine 1994;24(5):331-363.
* Hripcsak G, Ludemann P, Pryor TA, Wigertz OB, Clayton PD. Rationale for the Arden Syntax. Computers and Biomedical Research 1994;27:291-324.
* Jenders RA, Hripcsak G, Sideli RV, DuMouchel W, Zhang H, Cimino JJ, Johnson SB, Sherman EH, Clayton PD. Medical decision support: experience with implementing the Arden Syntax at the Columbia-Presbyterian Medical Center. Proc AMIA Symp 1995;:169-73.
* Johansson B, Shahsavar N, Åhlfeldt H. Database and knowledge base integration - a query mapping method for Arden Syntax knowledge modules. Meth Inform Medicine 1996;35:302-308.
* Karlsson D, Ekdahl C, Wigertz O, Shahsavar N, Gill H, Forsum U. Extended telemedical consultation using Arden Syntax based decision support, hypertext and WWW technique. Meth Inform Med 1997;36:108-114.
* Wigertz O, Hripcsak G, Shahsavar N, Bågenholm P, Åhlfeldt H, Gill H. Data-driven medical knowledge-based systems based on Arden Syntax. In Knowledge and decisions in health telematics. Barahona P and Christensen JP eds. Amsterdam: IOS Press, 1994;126-131.

**Q: Why should I use Arden Syntax rather than C, C++, Visual Basic, JAVA, or any other programming language or formalism that already exists? What are the advantages of the Arden Syntax?**

A: The Arden Syntax was specifically developed for health care applications and for embedding MLMs into clinical information systems. Its suitability can be further illustrated by the following features:

* *The target user is a clinician:* The Arden Syntax is not a full-feature programming language. It does not include the complex structures common in many other programming languages. MLMs are meant to be written and used by clinicians with little or no training in programming.
* *Explicit links to data, trigger events and messages to the target user:* The Arden Syntax was built to be embedded in existing clinical information systems. It clearly defines the hooks to clinical databases as well as how an MLM can be called (evoked) from a trigger event.
* *Time functions:* Almost all medical knowledge somehow involves stating the time when an event has occurred. Therefore, Arden Syntax defines that every data element and every event has a data/time stamp that is clinically significant. The Arden Syntax contains many time functions (explicitly defined) to help users handle date and time in MLMs. Arden Syntax defines the notion “duration” (e.g., year, month, week ) and shows how the user can check for example if an event has occurred in the last two days. In any other language, these definitions would be dependent on the person implementing the MLM, but Arden Syntax defines them explicitly.

**Q: What are the limitations of the Arden Syntax?**

A: A problem that occurs with any form of clinical knowledge representation is the need to interact with a clinical database in order to provide alerts and reminders. Since database schemata, clinical vocabulary and data access methods vary widely, any form of encoding of clinical knowledge (such as an MLM) must be adapted to the respective institution in order to use the local clinical repository. This hinders the sharing of knowledge. Due to the fact that Arden Syntax is the only standard for clinical knowledge representation, this problem is associated with Arden Syntax, but it is not unique to it.

The Arden Syntax explicitly isolates references to the local data environment in curly braces "{}" in an MLM, so this is sometimes called the "curly brace expressions problem". Efforts are underway in the HL7 Arden Syntax Work Group to help solve this problem, but this issue cannot be addressed by the Work Group alone; it requires industry-wide standardization of data access/storage.

Another potential limitation of the Arden Syntax is that it does not explicitly define notification mechanisms for alerts and reminders. Instead, this is left to local implementation and is–like database queries–expressed by curly brace expressions in an MLM. Explicit notification mechanisms in the Arden Syntax itself may be part of a future edition.

**Q: Some vendors refer to their systems as “compliant” with the Arden Syntax standard. What can I understand by that?**

A: When using an Arden-Syntax-compliant system, the user is able to create, import, customize, or otherwise implement MLMs without the need for vendor or system developer intervention. Additionally, the user can take an MLM from any institution, alter the content of the curly brace expressions, and make other related adjustments; the resultant MLM will be able to compile and execute at the user's institution.

**Q: What are the future plans for the Arden Syntax?**

A: The HL7 Arden Syntax Work Group is currently working on the Arden Syntax version 2.10. The new version will provide guidance on a standard data model that can be used by the Arden Syntax as well as incorporate the XML version of the standard in its normative section. The group is further working on an implementation guide that will show how to use the Arden Syntax to address various challenges in delivering clinical decision support.

# References

* [1] Hripcsak G. Writing Arden Syntax medical logic modules. Computers in Biology and Medicine 1994;24(5):331-363.
* [2] Hripcsak G, Ludemann P, Pryor TA, Wigertz OB, Clayton PD. Rationale for the Arden Syntax. Computers and Biomedical Research 1994;27:291-324.
* [3] Jenders RA, Hripcsak G, Sideli RV, DuMouchel W, Zhang H, Cimino JJ, Johnson SB, Sherman EH, Clayton PD. Medical decision support: experience with implementing the Arden Syntax at the Columbia-Presbyterian Medical Center. Proc AMIA Symp 1995;:169-73.
* [4] Osheroff JA, Teich JM, Levick D, Saldana L, Velasco FT, Sittig DF, Rogers KM, Jenders RA. Improving Outcomes with Clinical Decision Support: An Implementer’s Guide, Second Edition. Chicago: Healthcare Information and Management Systems Society, 2012.
* [5] Samwald, Matthias, et al. "The Arden Syntax standard for clinical decision support: Experiences and directions." Journal of biomedical informatics 45.4 (2012): 711-718.
* [6] Kraus, Stefan, et al. "Integrating Arden-Syntax-based clinical decision support with extended presentation formats into a commercial patient data management system." Journal of clinical monitoring and computing (2013): 1-9.
* [7] Adlassnig, Klaus-Peter, Alexander Blacky, and Walter Koller. "Artificial-intelligence-based hospital-acquired infection control." Stud Health Technol Inform 149 (2009): 103-10.