A Light-Weight Component for adding Decision Support to Electronic Medical Records

Jens H. Weber-Jahnke
University of Victoria, B.C., Canada

Glen McCallum
Inter Tribal Health, B.C., Canada

Abstract

Clinical decision support systems (CDS) are seen as a significant benefit to electronic medical records (EMRs). The use of CDS systems has been shown to improve adherence to medical evidence and increase the rate of adoption of new evidence. Many open source EMRs from small vendors do not provide CDS, because of the complexity of building them. Moreover, existing CDS functions often use proprietary knowledge representation, impeding knowledge exchange and update. This paper reports on our work on encapsulating standardized CDS functionality in a re-usable component that is agnostic of any particular EMR product. The component has been integrated with a sample EMR system and evaluated with a set of preventive care guidelines.

1. Introduction

Research has shown that publishing evidence-based clinical practice guidelines has limited immediate impact on care delivery and it can take years before published evidence and resulting best practices become routine in clinical medicine. The application of best practice (evidence-based) is surprisingly low, with areas such as prevention having approx. 50% adherence to guidelines [1,2]. Embedding patient specific reminders at the point of care is one method to improve adherence to evidence [3]. Knowledge-based clinical decision support systems (CDS) are seen as a significant benefit to using electronic medical records (EMRs) in clinical medicine. The use of CDS systems has been shown to improve adherence to the medical evidence and increase the rate of adoption of new evidence, that is, it can improve the rate of knowledge transfer [4]. A recent systematic review described four key success factors for CDS systems [5]:

- automatic decision support provided as part of routine clinical workflow,
- providing recommendations for action (vs. simple assessment),
- decision support at the time and location of decision-making, and
- computer-based decision support.

In the review of studies that adhered to these four features, 94% showed a significant impact on practice. This study supports finding from previous overviews and reviews on key features and clinical impact of decision support [6].

Current CDS systems are typically coupled to particular EMR systems. This coupling allows them to influence the clinical workflow and to provide alerts and reminders to the clinicians at various points in the delivery and documentation of care (e.g., ordering a lab test, prescribing a medication). However, the monolithical integration of CDS functionality in EMR systems also complicates their realization significantly, since CDS functionality is quite complex. Particularly, open source systems and small vendors would benefit from a standards-conform, EMR system agnostic CDS component that can simply be re-used in various systems.

This paper reports on the development and evaluation of such a component, which has now been made available in open source. Section 2 defines the interface of the CDS component including the data format used. Section 3 deals with issues of guideline represen-tation and their interaction with clinical data. Section 4 reports on the compilation of standard guidelines to a platform dependent format using a public domain expert system shell. Section 5 offers some preliminary experiences with running and testing the CDS component. Related work is discussed in Section 6 and we offer conclusions on Section 7.

2. Interaction interface

One of the most important questions to be decided during the realization of a re-usable CDS component is the design of the interface between EMR system and CDS component. The interaction protocol should be as simple as possible to facilitate adoption by many different EMR system vendors and projects.

The CDS component is to make as little assumptions about the EMR system as possible in order to achieve *lose coupling* and reusability. Consequently, the CDS system is not able to directly query the native database or data structures underlying the EMR system. Rather, we need to rely on a core patient summary in a standardized format to be sent to the CDS component for the purpose of analysis. This concept is analogous to the *virtual medical record* suggested by Johnson et al. [7].

One challenge is to determine exactly what data elements should be considered in this core patient summary. If the summary is defined too comprehensive, it may put too many requirements on the EMR system to interface with the CDS components. Moreover, large data volume may decrease the speed of invoking the CDS function to an unacceptable level. On the other hand, if the summary is too small, the usefulness of the CDS system may be decreased, e.g., cholesterol screening may require reviewing up to ten years of lab data).

One solution would be to define a multi-step protocol, which allows the CDS component to "*ask for more data*", if required during the analysis. However, care must be taken to keep the protocol simple in order to lower the cost of integrating the CDS component into EMR systems, i.e., to facilitate adoption as much as possible. We have defined a stateless, optional multi-step protocol to serve this purpose (Fig. 1).

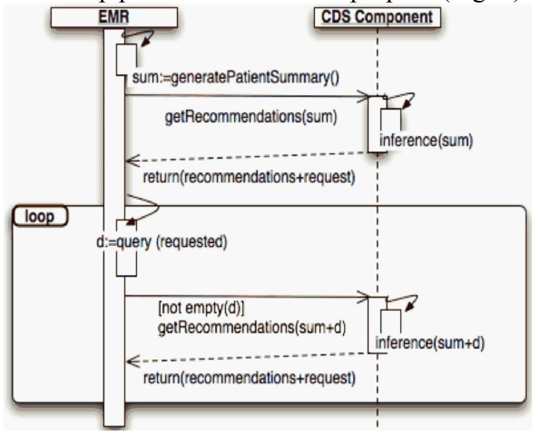

EMR
CDS Component
sum:=generatePatientSummary()
getRecommendations(sum)
inference(sum)
return(recommendations+request)
loop
d:=query (requested)
[not empty(d)]
getRecommendations(sum+d)
inference(sum+d)
return(recommendations+request)

Fig. 1. EMR <-> CDS protocol

The EMR system first generates an initial patient summary to be sent to the CDS component, including the data contained in Table 1. The CDS system uses coded clinical best-practice guidelines to produce *recommendations* based on this summary, including the information in Table 2. Attached to the recommendation results may be an optional request for additional data, which may be used in further guidelines. The EMR may or may not honour this request for additional data. If it does, it sends an extended patient summary to the EMR system. The design decision to resend the entire extended patient summary rather than just the additionally requested data simplifies the protocol, since no notion of state is necessary, i.e., we would have to add the concept of *sessions* otherwise. This protocol also has the advantage of allowing the EMR system to add extended data even to its initial call of "*getRecommendations*".

CDA Section	Description
Demographics	gender, birth date, marital status
Active Medical Problems	ICD-9-CM codes and diagnosis date for all active problems
Past Medical History	ICD-9-CM codes and diagnosis for all past problems
Surgical History	ICD-9-CM procedure codes
Current Medications	Health Canada Drug Product Database codes (DINs)
Allergies and Adverse Reactions	ICD-9-CM codes for allergies and adverse reactions
Family History	ICD-9-CM codes and diagnosis for family problems
Immunizations	British Columbia Center for Disease Control (BC CDC) codes
Social History	ICD-9-CM codes for such problems as smoking, etc
Recent Laboratory Data	LOINC codes for laboratory test

Tab. 1. Patient Summary Document

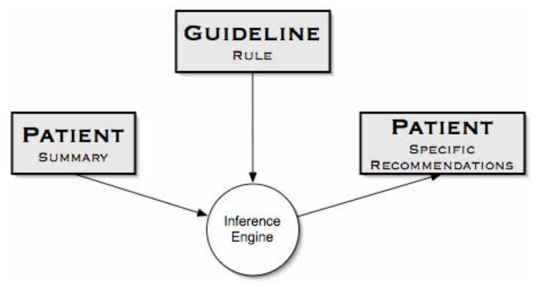
Element name	Description
guideline title	Title of the guideline
guideline encoding version	Encoding version number for the guideline
guideline citations	Reference for the guideline
guideline domain expert	Name of the clinical and informatics expert/specialist
guideline encoder	Name of the guideline encoder (who translated the guideline)
guideline encoding institution	Name of the institution that facilitated guideline encoding
guideline last encoded modification	The last time the guideline was encoded
guideline recommendation subject	Human readable title of the specific recommendation generated in the guideline
guideline recommendation instruction	Recommended response to the detected condition
guideline recommendation conclusion	The main conclusion statement of the guideline
guideline recommendation urgency	An integer from 1-99 ranking the urgency.

Tab. 2. Recommendations Document

As Tables 1 and 2 indicate, we have adopted the HL7 Clinical Document Architecture (CDA) as the data exchange format between EMR and CDS. CDA is an ANSI-approved HL7 standard for representing clinical documents in XML-format [8]. CDA documents consist of several sections, which may contain coded as well as unstructured data. In our case, we require fully coded data, in order to enable automatic guidelines processing. We currently support several coding terminologies used in the Canadian/B.C. context, including ICD-9 (International Classification of Diseases codes, rev. 9), Logical Observation Identifiers Names and Codes (LOINC), BC Centre for Disease Control Codes (BC-CDC), and Health Canada Drug codes. The set of coding terminologies used is extensible and we are currently working on integrating new ones, including SNOMED-CT.

We have chosen the CDA as the data exchange format because it is an international standard and it has been adopted by the Canadian information architecture standard [9] and used in several initiatives on the provincial level (e.g., the electronic medical summary standard, www.e-ms.ca). Many EMR vendors and open source systems already support import and export of CDA documents. This will lower their cost of interfacing with the CDS component.

We also use the CDA to format the recommendations returned from the CDS component. Each recommendation contains the elements listed in Table 2. For details about the structure of the two CDA documents, we refer to [10]



3. Guideline representation

The way in which practice guidelines (clinical knowledge) are coded within the CDS does not directly impact its functionality, but it greatly impacts its maintainability. It is possible to code guidelines in any general-purpose language. However, this approach greatly impedes knowledge exchange and maintenance. Dedicated clinical guideline formalisms (CGFs) have been developed to better support this aspect.

3.1 Selecting a format for representing executable clinical guidelines

An overview and comparison of some of them (GLIF, EON, PROforma, PRODIGY, Abru, GUIDE) can be found in [12]. We have studied these approaches in addition to one other approach (Arden Syntax) [11].

The criteria used for evaluating the CGF candidates are:

- **Standardization** - the CGF should be defined in a precise standard that is being actively maintained. Adopting a standards-based CGF would facilitate the exchange and maintenance of guidelines.
- **Decision-centric** - Guidelines may represent two aspects, namely medical knowledge and organizational knowledge [13]. Medical knowledge deals with medical decisions rules that generate alerts and recommendations. In other words, they define “what to do”. Organizational knowledge deals with defining and enacting clinical workflows over time, i.e., the “how to do it” and the “who does it”. Some CFGs are more appropriate for formalizing medical knowledge, while others are geared for enacting organizational knowledge. We concentrate on encoding medical knowledge because enacting organizational knowledge normally requires a tighter integration with the EMR workflow, which would put complicate the EMR-CDS interface.
- **Proven** – Some CGFs still seem to lack proper evaluation in practice. We did not want to be a first-time implementer of a novel CGF to avoid immaturity-related difficulties.
- **Free license** – One of our goals was to create a freely available open source CDS component. Therefore, we require a CGF with a free license.

Our evaluation resulted in selecting Arden Syntax [11] as our guideline coding format. Arden Syntax [11] is the only CGF that has been accepted as an official standard by ASTM, ANSI, and HL7. Arden Syntax is a decision-centric, rule-based formalism following the Event-Condition-Action paradigm.

3.2 Guideline representation in Arden

The core unit of formulating guidelines in Arden Syntax is the "Medical Logic Module" (MLM). Each MLM contains enough information to make a single medical decision, e.g., "*Should this patient be on Aspirin?*", "*Does this patient require Tetanus vaccine?*", etc. Arden Syntax is a textual language and its syntax has formally been defined while its semantics lacks a formal definition. Still, being decision-centric, interpreting MLMs is intuitive and for the most part, well described with natural language, without much room for ambiguity. Arden Syntax is a mature and actively maintained standard. With its first version released in 1992 there are several Arden-based CDS systems in existence today, e.g., Eclipsys, McKesson, Siemens. Arden Syntax is freely available without licensing fees. A more detailed discussion of the evaluation and comparison with other CGF candidates is beyond the scope of this paper and can be found in [14].

Every MLM has three sections:

1. The **Maintenance** section contains general meta-information about the guideline, such as its title, version number, author, status, etc.
2. The **Library** section contains scientific meta-information about the guideline, e.g., a natural language explanation, references to literature, keywords, purpose etc.
3. The **Knowledge** section contains the actual guideline algorithm.

The contents of the first two sections (Maintenance and Library) are generally not machine interpreted, but they provide data returned with the recommendation (cf. Table 2). The Knowledge section contains the actual algorithm. Consider the "Diabetes A1C guideline" in Fig. 3 as an example. It codes the knowledge that an A1C lab test, which gives an indication of a patient's blood sugar levels over the past 2-3 months, is due in diabetics every three months. If the A1C result is above a target range of 7%, then the clinician should be reminded and consider changing the current diabetes management. The data input consists of: 1) a list of all the types of diabetes a patient has been diagnosed with, and 2) the value, unit, and date of the last A1C lab test.

Fig. 4 shows the algorithmic represent-ation of the sample guideline in Arden Syntax. Only the Knowledge section is shown, including its major *"slots"*: data, logic, and action. The data slot defines variables by reading input data from our patient summary, the logic slot contains the actual condition and the action slot defines the recommendation. In our case, actions are always creating an XML entry to be included in the CDA document sent back to the EMR.

Note that the guideline in Fig. 4 has been simplified for presentation in this paper. The full guideline is contained in [14].

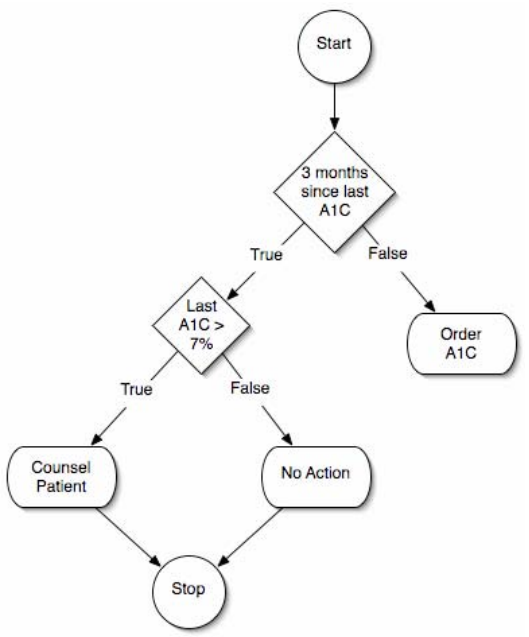


Fig. 3. Diabetes A1C Guideline

```
knowledge:
  data:
    let diabetes be read exist {…};
    let latest_a1c_value be read {…};
    let latest_a1c_unit be read {…};
    let latest_a1c_date be read {…};
  logic:
    […]
    if ( (now - latest_a1c_date) < 3 months)
     then if ( (latest_a1c_value > 7) AND
               (latest_a1c_unit = "%"))
            then
                patient_above := true;
                conclude true;
            else
                conclude false;
            endif;
        else
            patient_due := true;
            conclude true;
        endif;
    action:
      if patient_above then
            write    "<?xml …>";
      elseif patient_due then
            write    "<?xml …>";
        endif;
    urgency: 60;;
```

Fig. 4. Arden Syntax representation of Diabetes A1C Guideline

3.3 Linking guidelines to data

One of the simplifications done in Fig. 4 was to omit the data query statements inside the curly braces "{…}" in the data section of the MLM. Since the syntax for such data queries depends on the nature of the data source, it is not defined in the Arden Syntax standard. In our case, we are using CDA (XML) documents to communicate a "virtual medical record" between the EMR and the CDS. Therefore, we have chosen the XML query language XPath [15].

Fig. 5 gives an example for querying the CDA patient summary for the presence of diabetes. The XPath query navigates the hierarchical structure of the XML up to the section on *active problems* that have been observed by physicians (provider code 114504). Several ICD-9 codes may be used to enter different classifications of diabetes. The query matches ICD-9 codes 648.0 (diabetes mellitus), 648.8 gestational diabetes, 790.29 (abnormal glucose), and 250.1-250.9 (sub-classifications of diabetes mellitus).

```
Let        diabetes        be        read
{ClinicalDocument/structuredBody/
activeProblems/section[code/@code="114504"]
/entry/observation/code[starts-with(@code,
"250.") or @code="648.0" or @code="648.8" or
@code="790.29"]/@code}
```

Fig. 5. Query on CDA Document

4. Guideline compilation

We implemented the guideline execution mechanism based on CLIPS expert system shell [16]. CLIPS was developed by NASA at the Johnson Space Centre in the 1980s. It is a forward-chaining, multi-paradigm, expert system shell that can easily be embedded in other programs. It is public domain software and binary distributions are available for all major operating systems. In order to realize the guideline execution in CLIPS, we needed to perform two tasks:

1. We created a **library** of CLIPS functions that implement the semantics of the Arden Syntax operators.
2. We created a **compiler** that would translate MLMs into CLIPS rules.

4.1 Implementing Arden operators

Many Arden Syntax operators readily translate to CLIPS operators. However, difficulties arise with Arden's temporal operators (e.g., *latest* in Fig. 4) and corresponding temporal data types (e.g., *time* and *duration*). Arden allows durations to be specified in various ways, e.g., as months, years, seconds, etc. All these cases have to be considered in the CLIPS library. Fig. 6 illustrates this for the (simplified) operator function that subtracts durations from times.

Implementing the temporal operators for Arden requires the implementation of a collection of time conversion functions. One example is the conversion of absolute points in time expressed in <day/month/year> to relative counts of seconds since 01/01/1800. (The Arden standard mandates that dates are to be supported back to this date.) The two functions *secondsSinceBase* and *secondsSinceBaseToTime* used in Fig. 6 are examples for such conversion functions in our Arden library.

Temporal computations are complex and we discovered ambiguities in the Arden Syntax standard specification while implementing CLIPS library functions. We describe three such ambiguities here. Others are included in [14].

```
(deffunction timeMinusDuration (?time ?duration)
 (bind ?year (sub-string 1 4 ?time))
 (bind ?month (sub-string 6 7 ?time))
 (bind ?day (sub-string 9 10 ?time))

 ; duration is in months
 (if (eq (durGetUnit ?duration) "months") then

  ; months / 12 -> years to subtract
  (bind ?years (- ?years (/(durGetNum ?duration) 12)))
  ; leftover months
  (bind ?months (- ?months (mod (durGetNum ?duration)
12)))

  ; have to roll back a year? (negative month)
  (if (< ?months 1)
    (bind ?years (- ?years 1))
    (bind ?months (+ ?months 12))
  )
  (return (str-cat ?years "-" ?months "-" ?day))

  ; duration is in seconds  else
  ; get the number of seconds since 1800-01-01
  (bind ?oldSecondsSinceBase (secondsSinceBase ?time))

  ; subtract seconds duration
  (bind ?newSecondsSinceBase (- ?tseconds (durGetNum
?duration)))

  ; convert seconds since 1800-01-01 back to a time
  (return                    (secondsSinceBaseToTime
?newSecondsSinceBase))
 )
)
```

Fig. 6. CLIPS implementation of Arden temporal subtraction operator

- *Negative duration*. Arden has operations for subtracting one time from another time, resulting in a duration. For example, 2006-02-21 – 2006-01-21 = 1 month. However, what happens if the left-hand side date is younger than the right hand side date? Do negative durations exist? If yes, what is their semantics? Does "minus one month" mean one month ago? The Arden standard is mute on this question and leaves it up to the implementer to decide. Consequently, different Arden implementa-tions may interpret identical guidelines differently.
- *Operator Conflicts*. With respect to durations and arithmetic there are a series of unary operators associated with duration units (years, months, days, hours, minutes, seconds, [11], page 50). These operators convert year durations to month durations, and days, hours, and minute durations to second durations. Consider this example: The expression "2 years > 24 months" is evaluated by first applying the unary *years* operator to yield "24 months > 24 months". Then the binary operator > is applied. This precedence rule appears however to contradict with another part of the specification [11], page 48), which shows the example of subtracting a duration from another duration (3 days – 2 days = 1day). If the "days" unary operator was applied before the binary operator "-", then the result should be given in seconds. It appears that the Arden Syntax designers implicityly assume that seconds are converted back to days. We made the decision to code such an automatic back-conversions step. In the example of our operator implementation in Fig. 6 this conversion is performed by function *secondsSinceBaseToTime*, converting seconds back to time (day, month, year).
- *Overflow and underflow.* At several locations the Arden specification states the rule "In case of underflow/overflow return null." However, there is no precise definition as to what constitutes such a situation. It is left up the the implementer (the chosen computer architecture, selected data types etc.) to determine the occurrence of overflow or underflow events.

4.2 Compiling MLMs to CLIPS

We have constructed a compiler that translates MLMs to CLIPS rules using the SableCC compiler construction framework. Figure 7 explains the compilation process for a simple example guideline (Tetanus booster). Each MLM maps to two CLIPS structures, a CLIPS Rule and a CLIPS Function. The rule is shown at the bottom of Fig. 7 starting with "`(defrule tetboost_body…`". It has a left-hand side and a ride-hand side that executes only if the left-hand side evaluates to true. The data section of the MLM maps to the left-hand side of the generated CLIPS rule (arrow line 1). The right-hand side of the rule contains the logic portion of the MLM (line 2). Therefore, if all data elements are present in the patient summary then the right-hand side of the rule (logic portion of the MLM) executes. In this fashion, MLMs are data-driven, i.e., they are invoked when a Patient Summary CDA document arrives and this document contains the appropriate data entries. (The actual XPATH query is not executed in CLIPS but in Java, since CLIPS lacks proper XML/XPATH libraries.)

The action slot of the Arden Syntax MLM translates to a function in CLIPS (arrow line 3). This function (beginning with `(deffunction tetboost_action…`") is called if the logic portion of the MLM concludes true. It uses the library of operators we have implemented for the corresponding Arden operators. For example the *durgtdur* operator is an implementation of the temporal greater operator ">" used to compare durations.

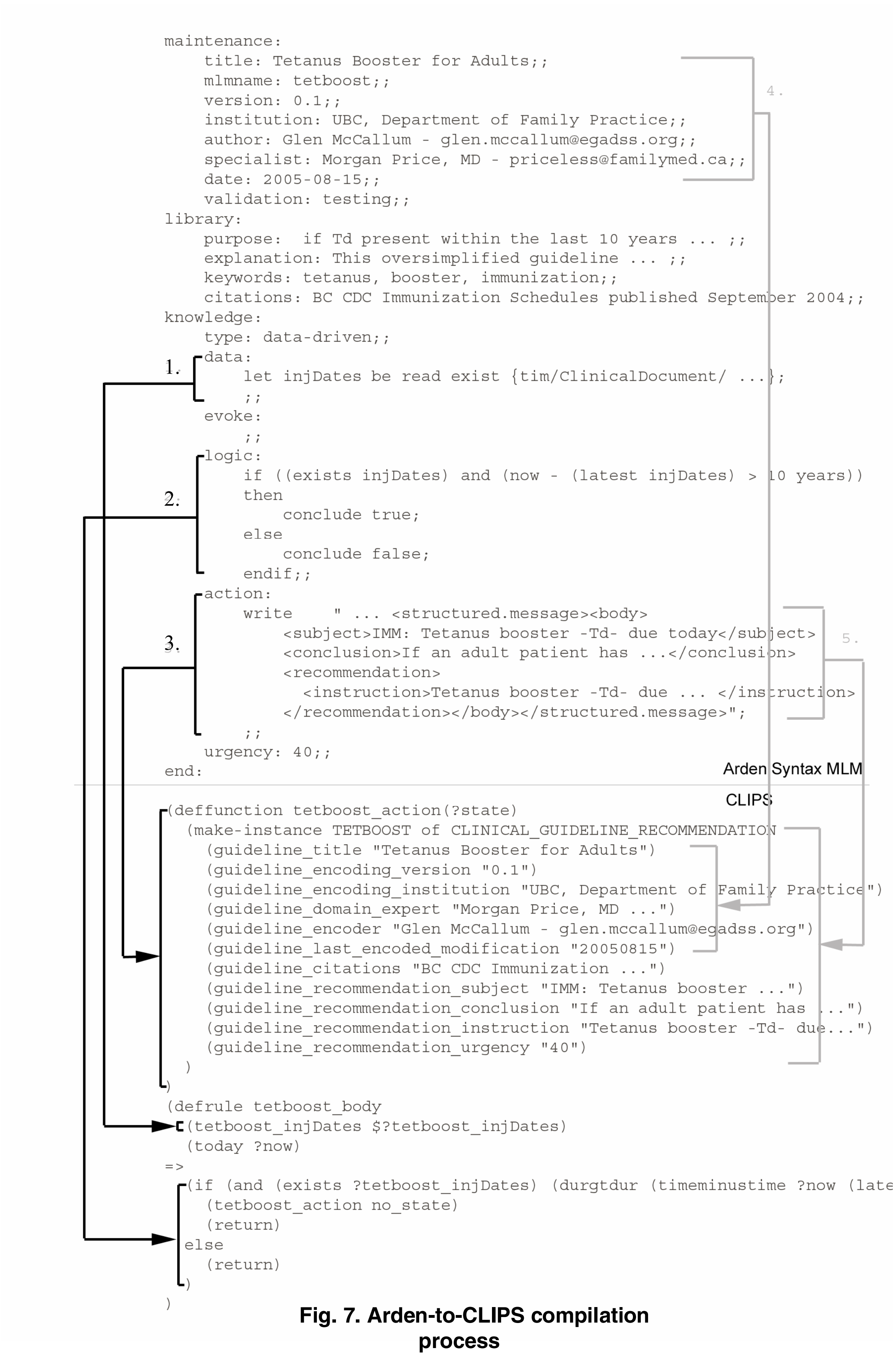
Each Arden "Write" statement translates to a CLIPS *make-instance* statement of a CLINICAL_GUIDELINE_RECOMMENDATI ON (line 5). Some of the information in the recommendation instance comes from the maintenance section of the MLM (line 4).

5. Evaluation

5.1 Correctness and performance

We have implemented the CDS component as a standard Web service component within the open source Tomcat application server. Choosing the standard Simple Object Access Protocol (SOAP) allows us to be completely platform-independent. The following steps are performed during the operation of the CDS component:

1. Compile MLMs and load generated CLIPS rules. (at startup)
2. Wait for arrival of patient summary
    - Apply XPath queries in *data* sections of MLMs and assert results as CLIPS facts



```
maintenance:
    title: Tetanus Booster for Adults;;
    mlmname: tetboost;;
    version: 0.1;;
    institution: UBC, Department of Family Practice;;
    author: Glen McCallum - glen.mccallum@egadss.org;;
    specialist: Morgan Price, MD - priceless@familymed.ca;;
    date: 2005-08-15;;
    validation: testing;;
library:
    purpose:  if Td present within the last 10 years ... ;;
    explanation: This oversimplified guideline ... ;;
    keywords: tetanus, booster, immunization;;
    citations: BC CDC Immunization Schedules published September 2004;;
knowledge:
    type: data-driven;;
    data:
        let injDates be read exist {tim/ClinicalDocument/ ...};
        ;;
    evoke:
        ;;
    logic:
        if ((exists injDates) and (now - (latest injDates) > 10 years))
        then
            conclude true;
        else
            conclude false;
        endif;;
    action:
        write    " ... <structured.message><body>
            <subject>IMM: Tetanus booster -Td- due today</subject>
            <conclusion>If an adult patient has ...</conclusion>
            <recommendation>
              <instruction>Tetanus booster -Td- due ... </instruction>
            </recommendation></body></structured.message>";
        ;;
    urgency: 40;;
end:
```

```
(deffunction tetboost_action(?state)
  (make-instance TETBOOST of CLINICAL_GUIDELINE_RECOMMENDATION
    (guideline_title "Tetanus Booster for Adults")
    (guideline_encoding_version "0.1")
    (guideline_encoding_institution "UBC, Department of Family Practice")
    (guideline_domain_expert "Morgan Price, MD ...")
    (guideline_encoder "Glen McCallum - glen.mccallum@egadss.org")
    (guideline_last_encoded_modification "20050815")
    (guideline_citations "BC CDC Immunization ...")
    (guideline_recommendation_subject "IMM: Tetanus booster ...")
    (guideline_recommendation_conclusion "If an adult patient has ...")
    (guideline_recommendation_instruction "Tetanus booster -Td- due...")
    (guideline_recommendation_urgency "40")
  )
)
(defrule tetboost_body
  (tetboost_injDates $?tetboost_injDates)
  (today ?now)
=>
  (if (and (exists ?tetboost_injDates) (durgtdur (timeminustime ?now (late
    (tetboost_action no_state)
    (return)
  else
    (return)
  )
)
```

Fig. 7. Arden-to-CLIPS compilation process

- Run CLIPS rules (Rules operate on patient facts to produce new recommendation facts).
- Export recommendation facts to CDA results document

3. Reset CLIPS to prepare for next patient document (Removes all facts but leaves rules loaded)
4. Go to (2)

On a Pentium X CPU, a typical SOAP service time for parsing a patient summary document, executing 5 guidelines, and generating the recommendations document averages at 2-3 seconds. The resulting recommendations have been reviewed for correctness by a medical doctor on our team. A simple installer package and a Web based demo including the sample guidelines are available at www.egadss.org.

5.2 Adoptability claim

Evaluating our claim that the designed CDS component's light-weight design facilitates adoption ideally requires a long-term empirical study of the adoption rate of this new software in order to be conclusive. Since such a study will likely take years, the argument made in this paper intends to extrapolate on experiences with prior empirical studies. We attempt to show that we have followed established best practice design rules, which have proven to result in highly adoptable and integratable component designs. Hess and Humm have studied, compiled and published rules for designing loosely coupled, highly maintainable service components [20]. These rules have been validated in empirical studies over more than five years and 100 person/years within the *Quasar* project (Quality Software Architecture) at sd&m, a European software company. The rules and their manifestation in our project are described in the following.

- *R1: Technology Neutral - Service operations should expose domain-oriented functions only. They should not reveal any details about the implementation technology.*

We selected the CLIPS expert system shell developed by NASA as the inference engine. However, this choice of technology does not shine through to the service client; even operations that enable knowledge maintainers to update clinical guidelines do not expose any CLIPS-based details.

- *R2: Reference-free - Parameters of service operations should not contain references to internal objects.*

We have chosen that clients should provide all input data for CDS in a self-contained CDA document (the patient summary). This document does not contain any references to additional data objects residing in the EMR‘s repository. Likewise, the response created by the CDS is in form of a self-contained CDA document (the recommendation document), which does not contain any further reference to CDS objects.

- *R3: Normal form - Elementary service operations (those which are not composed of other service operations) should be in normal form, i.e., they should be complete and free of redundancy.*
- *R4: Coarse Granular - Third-party component service invocations (remote procedure calls) may be expensive in terms of latency and communication overhead. Therefore service operations should be of coarse granularity (few calls to get desired effect).*

In our interface design, only one single invocation of operation "getRecommendation" is necessary to get the recommendations. This invocation requires an extendible patient summary as the input parameter, i.e., a data parameter than can contain more or less information based on the specific context of the service invocation.

- *R4: Idempotent - Service operations should be idempotent, i.e., each time they are invoked, they should have the same effect for the same input parameters.*
- *R5: Context-free - Operations should not have knowledge (and not make any assumptions) about the context they are called in.*

The purpose of these rules is to minimize the coupling of components with respect to their state. We have implemented these rules since our CDS is completely stateless; each call to "getRecommendations" is idempotent and depends only on the patient summary provided in the current input argument. In other words, the CDS is memoryless. There is no concept of a "query session" that involves several calls between EMR and CDS. Our CDS does not persist any clinical data. Therefore, there is no state-based coupling between CDS and EMR.

In addition to our theoretical argument, we have done a practical proof-of-concept integration of our CDS with an existing EMR system (OpenTAPAS, www.opentapas.org). OpenTAPAS had pre-existing capabilities for generating and importing CDA documents conforming to the e-MS standard (www.e-ms.org). Modifying OpenTAPAS to be able to generate our patient summaries took one person/work day. (We needed to add laboratory history to the the generated CDA document. This information existed in the OpenTAPAS repository but was not included in the generated e-MS CDA document.) Building a rudimentary Web Viewer to display the recommendations from the CDS with a CDA-based style sheet took one more person day. A more intelligent integration of alerts and reminders generated by EGADSS into the OpenTAPAS clinician messaging system took approximately one person/week. We believe that this practical integration further demonstrates that the CDS design facilitates re-usability and adoption.

6. Related work

Ciccarese et al. have developed *NewGuide*, a system for developing, managing and executing clinical guidelines based on a flow-chart like representation [18]. NewGuide also supports a SOAP-based interface to add value to legacy EMR systems. However, NewGuide guidelines are workflow-centric and their interface is stateful and more complex. It is not clear what level of implementation and evaluation could be achieved with the NewGuide approach.

A similar approach has been taken by Wang et al. with the Guideline Execution Engine GLEE [19]. It uses the workflow-centric visual language called GLIF3 and requires a more complex, interaction protocol with the EMR, keeping track of the states of tasks, suspensions and continuations.

Both, NewGuide and GLEE, use non-standard guideline formalisms, which may impede knowledge exchange. With their workflow-centric nature, these systems are more powerful, but also significantly more complex to deploy and integrate with existing EMRs. Neither of these systems are available as off-the-shelf components for integration by third-parties.

7. Conclusions

Many existing open-source or small vendor EMR solutions would benefit from adding a decision-support function for better adherence and dissemination of best-practice guidelines. We have developed a standards-based, light-weight and reusable CDS component for this purpose and released it into open source. Our system is the only Arden Syntax CDS available in open source to date. In contrast to related projects, our declared goal was to keep the interface between CDS component and EMR as simple as possible, in order to ease integration. We have evaluated the CDS component with five sample guidelines, three immunization guide-lines: (Td Primary, MMR, Pneu C7) and two Chronic Disease Management guidelines (Diabetes A1C, Diabetes Hypogly-cemia). We have also performed a proof-of-concepts integration with the openTAPAS open-source EMR [17]. The entire software has been released on sourceforge, including the five coded guidelines (www.egadss.org). A Web-based demo is also available. We will continue to evolve the system and provide support for a community of adopters and developers.

References

[1] J. Lomas et al., “Do practice guidelines guide practice? The effect of a consensus statement on the practice of physicians,” N Engl J Med, 321, 1989, pp. 1306-1311.

[2] J. M. Grimshaw, “Effect of clinical guidelines on medical practice: a systematic review of rigorous evaluations,” Lancet, 342, 1993, pp. 1317-1322.

[3] S. Weingarten, “Translating Practice Guidelines Into Patient Care - Guidelines at the Bedside,” 118, 2000, Am Coll Chest Phys, pp. 4-7.

[4] M. E. Johnston et al., “Effects of Computer-based Clinical Decision Support Systems on Clinician Performance and Patient Outcome: A Critical Appraisal of Research,” Ann Intern Med, 120, 1994, pp. 135-142

[5] K. Kawamoto et al., “Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success,” 330, 2005, Br Med Assoc, pp. 765.

[6] D. W. Bates et al., “Ten commandments for effective clinical decision support: making the practice of evidence-based medicine a reality,” J Am Med Inform Assoc, 10, 2003, pp. 523–530.

[7] Johnson PD, Tu SW, Musen MA, Purves I., “A virtual medical record for guideline-based decision support.”, Proc AMIA Symp. 2001;:294-8.

[8] R. H. Dolin et al., "HL7 Clinical Document Architecture, Release 2," JAMIA, 13, 2006, pp. 30-39.

[9] Infoway, EHRS Blueprint -- an interoperable EHR framework, Canada Health Infoway, Montreal, 2006.

[10] I. Bilykh, J. H. Jahnke, G. McCallum, and M. Price. "Using the clinical document architecture as open data exchange format for interfacing EMRs with clinical decision support systems". In Proc. of 19th IEEE Intl. Symp. on Computer-Based Medical Systems, pages 855–860. IEEE CS Press, 2006.

[11] ANSI, Arden syntax for medical logic systems, v2, American National Standards Institute, Tech. Rep. ANSI/HL7 Arden V2.1-2002, Washington, DC, 2002.

[12] M. Peleg et al., "Comparing computer-interpretable guideline models: a case-study approach," J Am Med Inform Assoc, 10, 2003, pp. 52-68.

[13] S. Quaglini et al., "Guideline-based careflow systems," Artif Intell Med, 20, 2000, pp. 5-22.

[14] G. McCallum. "EGADSS: A Clinical Decision Support System for use in a Service-oriented Architecture", MSc. Thesis, Computer Science, University of Victoria, BC, Canada p.148, (2006)

[15] J. Simeon, and P. Wadler, "The essence of XML," Proc. of the 30th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, 2003, ACM Press New York, NY, USA, pp. 1-13.

[16] G. Riley, "CLIPS: An expert system building tool," NASA, Washington, Technology 2001: 2nd National Technology Transfer Conference and Exposition, 2, 1991

[17] B. Barclay, M. Price, and J. H. Jahnke. Open technology assisted practice application suite. Intl. Conf. on Inform. Technology & Communication in Health (ITCH). University of Victoria, 2007.

[18] P. Ciccarese, E. Caffi, L. Boiocchi, S. Quaglini, and M. Stefanelli. A guideline management system.

Medinfo, 11(Pt 1):28–32, 2004.

[19] D. Wang, M. Peleg, S. W. Tu, A. A. Boxwala, O. Ogunyemi, Q. Zeng, R. A. Greenes, V. L. Patel, and

E. H. Shortliffe. Design and implementation of the glif3 guideline execution engine. J Biomed Inform,

37(5):305–318, 2004.

[20] A. B. Hess, and M. V. Humm, "Regeln für serviceorientierte Architekturen hoher Qualität," Informatik Spektrum, 29, 2006, pp. 395.