J Clin Monit Comput (2014) 28:465–473
DOI 10.1007/s10877-013-9430-0

ORIGINAL RESEARCH

Integrating Arden-Syntax-based clinical decision support with extended presentation formats into a commercial patient data management system

Stefan Kraus · Ixchel Castellanos · Dennis Toddenroth · Hans-Ulrich Prokosch · Thomas Bürkle

Received: 1 August 2012 / Accepted: 9 January 2013 / Published online: 26 January 2013

Abstract The purpose of this study was to introduce clinical decision support (CDS) that exceeds conventional alerting at tertiary care intensive care units. We investigated physicians' functional CDS requirements in periodic interviews, and analyzed technical interfaces of the existing commercial patient data management system (PDMS). Building on these assessments, we adapted a platform that processes Arden Syntax medical logic modules (MLMs). Clinicians demanded data-driven, user-driven and time-driven execution of MLMs, as well as multiple presentation formats such as tables and graphics. The used PDMS represented a black box insofar as it did not provide standardized interfaces for event notification and external access to patient data; enabling CDS thus required periodically exporting datasets for making them accessible to the invoked Arden engine. A client–server-architecture with a simple browser-based viewer allows users to activate MLM execution and to access CDS results, while an MLM library generates hypertext for diverse presentation targets. The workaround that involves a periodic data replication entails a trade-off between the necessary computational resources and a delay of generated alert messages. Web technologies proved serviceable for reconciling Arden-based CDS functions with alternative presentation formats, including tables, text formatting, graphical outputs, as well as list-based overviews of data from several patients that the native PDMS did not support.

S. Kraus (✉)
Center for Communication and Information Technology, University Hospital Erlangen, Erlangen, Germany
e-mail: stefan.kraus@uk-erlangen.de

I. Castellanos
Department of Anesthesiology, University Hospital Erlangen, Erlangen, Germany
e-mail: ixchel.castellanos@kfa.imed.uni-erlangen.de

D. Toddenroth · H.-U. Prokosch · T. Bürkle
Department of Medical Informatics, Biometrics and Epidemiology, Chair of Medical Informatics, Friedrich-Alexander-University Erlangen-Nuremberg, Erlangen, Germany
e-mail: dennis.toddenroth@imi.med.uni-erlangen.de

H.-U. Prokosch
e-mail: ulli.prokosch@imi.med.uni-erlangen.de

T. Bürkle
e-mail: thomas.buerkle@imi.med.uni-erlangen.de

Keywords Arden Syntax · Clinical decision support · Clinical event monitoring · Patient data management system · Intensive care unit

1 Introduction

The Arden Syntax for medical logic systems [1] is a language for encoding medical knowledge. It is an HL7 standard that is supported by a number of commercial clinical information systems. We integrated clinical decision support (CDS) functions based on the Arden Syntax into a commercial patient data management system (PDMS) to supply additional functions, like monitoring laboratory values, score calculations, information retrieval for billing purposes, and even treatment guidelines.

This paper concentrates on the technical aspects of integrating CDS in a typical, commercially available PDMS, and specifically addresses these research questions:

- Like many other clinical information systems the used commercial PDMS is a black box system, insofar as it does not provide interfaces required for integrating CDS functions. Is it nevertheless possible to integrate a full featured technical platform to execute Arden Syntax knowledge modules?

- Some of the requirements of our clinical users such as tabulated and graphical outputs clearly go beyond the typical scope of the Arden Syntax. In particular there was a request for list-based processing and for mechanisms to present generated information. Is it expedient to implement such mechanisms directly in Arden Syntax and what are the potential benefits?

2 Environment

The University Hospital of Erlangen (Universitätsklinikum Erlangen, UKER) is a tertiary care university hospital with 1.400 beds. In 2006 UKER introduced a commercial PDMS (Integrated Care Manager, ICM®, Dräger Medical, Lübeck, Germany) in an interdisciplinary operative intensive care unit (ICU) [25]. The PDMS has since been rolled out at eight ICUs altogether, covering about 100 beds of surgical, neurosurgical, medical and paediatric intensive care.

At an early stage during the rollout clinical users asked for additional capabilities in terms of tools for ICU scoring, monitoring of critical data, and various kinds of information retrieval for the purpose of decision support. Especially scores with a specific relevance for intensive care such as PRISM score, MELD score or RIFLE/AKIN score which simultaneously consider the constellation of different patient parameters were required.

We decided to search for a flexible and expandable technical solution based on available standards. Thus a development cooperation with the PDMS vendor was initiated to integrate the PDMS with a commercial Arden environment (Arden Syntax IDE®, Medexter Healthcare, Vienna, Austria).

3 Background

A clinical decision support system (CDSS) is a computer program “designed to aid directly in clinical decision making, in which characteristics of individual patients are used to generate patient-specific assessments or recommendations that are then presented to clinicians for consideration” [2]. For decades these systems have been in use, and several systematic reviews have corroborated that they can enhance clinical performance [3–8]. The utilization of a CDSS requires the medical knowledge to be encoded in a representation format that can be processed by a computer. The Arden Syntax is such a format. Its design objectives were knowledge transfer between institutions and easy knowledge encoding [9]. In Arden, medical knowledge is structured in independent modules, called medical logic modules (MLMs), which usually contain sufficient knowledge to make one single clinical decision. For an introduction of how to write MLMs see [10].

Using Arden syntax presupposes a set of technical components: The first one is an Arden compiler to make MLMs executable, for example by translating Arden Syntax into a different programming language such as Java [11–14], C++ [15, 16], or stored procedures of the underlying database system [17, 18]. A later approach is using ArdenML [19] and extensible stylesheet language transformation (XSLT) instead of a classical compiler [20]. The second component is an Arden engine, which controls the execution of MLMs by reacting to clinical events. Compiler, engine and additional tools like MLM editors and management tools form the Arden environment.

MLMs are specifically designed for clinical event monitoring [21], typically running invisibly in the background unless they identify a predefined critical situation that causes an alert. Hripscak et al. [22] call event monitors “tireless observers, constantly monitoring clinical events”, and distinguish three basic types of events, depending on whether program execution is triggered by data entry (*data-driven*), by GUI events (*user-driven*), or executed periodically (*time-driven*). Integrating a clinical event monitor into an existing information system thus requires (at least) a trigger mechanism and a way to access data.

If an institution wants to adopt the Arden Syntax, it encounters two basic problems. The first one, called the *compiler problem*, is that building an Arden compiler is a difficult and resource-intensive task [19]. Recently both a commercial Arden environment [13] as well as an open source solution [23] have become available. The second one, referred to as the *curly braces problem*, is that data and message types of a clinical information system must be mapped onto Arden Syntax data types and vice versa (since the mapping is specified by the parameters inside a pair of curly braces). The designers of the Arden Syntax deliberately left this mapping process to the particular institution due to the lack of standardized interfaces and patient records [24].

4 Methods

Integrating CDS functions based on the Arden Syntax into an existing PDMS has to take into account the requirements of the clinical users and the technical properties of the interfaces to be connected. Therefore, these steps had to be taken:

- Analyzing the clinical and technical requirements.
- Creating a technical platform for executing MLMs by connecting the PDMS and the Arden Engine. This

comprised workarounds for improving interfaces, solving the curly braces problem and extending the system for a multi-ICU environment.

- Creating mechanisms for presenting generated information. We decided to perform this step completely in Arden Syntax.

4.1 Analyzing requirements

We had already integrated a basic CDSS prototype as a proof of concept before the beginning of the funded cooperation. This gave us a good impression of the clinical users' requirements that was expanded in periodic interviews and enabled us to create a blueprint of the system to build.

The technical requirements were determined by analyzing the interfaces of the PDMS and the Arden Engine. The Arden engine provides a generic interface named ArdenHostInterface to connect it to a clinical information system. We used some example code shipped with the Arden environment to examine the ArdenHostInterface. A set of MLMs were written and their interaction with the ArdenHostInterface was investigated. The PDMS provides an export interface ("exporter") that has already been extensively used in our hospital for other purposes like the generation of discharge letters, so we could analyze a large set of existing export jobs to investigate its function.

4.2 Connecting the PDMS and the Arden engine

On the PDMS side we had to provide the lacking data-driven trigger mechanism and a way to access patient data. As the exporter was the only available interface, we had to build workarounds. Data access was enabled by periodically replicating the required parts of the patient records into an external database that acts as a proxy for the PDMS patient database ("proxyDB"). We set up an export job on a dedicated export virtual machine (VM) that creates structured text files with patient data that are parsed by a self-written export handler that writes their content into the proxyDB. Event detection is carried out by comparing exports. Whenever the export handler detects a data item that has not been previously exported, it creates an entry in an event list that is finally sent to the Arden Engine. Another workaround was needed for providing microbiological data. The PDMS can only provide data that is insufficiently structured for automatic processing, so we duplicated the stream of HL7 messages at our communication server. A self-written parser processes these messages and writes their content into the proxyDB, preserving its structure.

On the Arden side we implemented the control and curly braces evaluation methods of the ArdenHostInterface. The Arden engine was integrated into an open source application server to provide a web service interface. The resulting system is called the Arden server. With regard to the curly braces problem we focussed on making the structure of the mapping clauses as simple as possible for the knowledge engineers in order to support the paradigm of "doctors as programmers" [26].

After successfully providing decision support on a first ICU, we expanded our system to a multi-ICU environment. A centralized export of patient data across multiple ICUs is not supported by the PDMS, so we built a partly distributed system. Hence, we provided one export VM with one export job and export handler on any ICU.

For user driven execution of MLMs we wrote a standalone mini web browser ("MLM viewer") and connected it to a GUI element ("Arden button") in the PDMS. We have ensured that the MLM viewer is always running in the foreground as long as it is used to avoid mistaken patient identities by viewer instances that could be forgotten in the background. We then built a web application with a button row and a display area to show MLM-generated output. A web service client was integrated that sends event messages to the Arden Server on button click.

4.3 Presenting generated information

Because all requirements regarding the presentation of results could be fulfilled using web technologies, we decided to completely implement all presentation mechanisms in Arden Syntax by writing presentation MLMs that generate HTML and JavaScript. We implemented MLMs for formatting fonts, generating tables from plain lists, generating more complex tables from description objects, expanding and collapsing elements by mouse click, and for generating scalable line plots from description objects. We created a separate MLM reserved for all object declarations that can be included in other MLMs. For the line plots we used an open source JavaScript library.

5 Results

We were able to integrate a full featured technical platform to execute MLMs by providing workarounds for data access and data-driven event notification. MLMs can be executed by the Arden Engine over a web service interface. We provided a web application shown in an MLM viewer for user-driven execution of MLMs. Generated results can be presented using a set of presentation MLMs. A total of 30 MLMs (presentation MLMs not included) are in clinical routine use at an interdisciplinary surgical and a pediatric ICU of the UKER. They can be divided into MLMs for data-driven monitoring of laboratory values (such as potassium and blood glucose), and MLMs for user-driven decision support (like formula calculations such as anion gap or ventilation volume, or calculation of score profiles). These 30 MLMs have been cumulatively invoked 161.400 times (84.3 % data-driven, 15.7 % user-driven by 144 distinct users) within one year, corresponding to a daily use of 442 invocations on average.

5.1 Analyzing requirements

The interviews showed that the use of alert boxes was clearly unrequested. The desired methods of communication with the system were emails and SMS messages for data-driven and time-driven MLMs and a graphic user interface started by a GUI button ("Arden button") for user-driven MLMs. Figure 1 shows the basic architecture of the intended system.

The required presentation mechanisms comprised text formatting, creation of tables and line plots, and the ability to expand and collapse arbitrary elements on mouse click. There was also a large requirement for list-based MLMs to simultaneously process information for numerous patients to generate overviews which the PDMS does not support.

The technical analysis of the PDMS exporter showed that it is capable of exporting patient data into arbitrary text formats and executing command line statements. It supports user-driven and time-driven, but not data-driven execution. There is no way to start an export from outside the PDMS.

The ArdenHostInterface is initially empty and must be implemented by the customer by providing control methods to operate the Engine externally and curly braces evaluation methods that are implicitly called whenever the control flow reaches a statement associated with a curly braces expression (e.g. a READ-Statement).

Therefore, the clinical and technical requirements involved following steps of implementation:

- Enabling access to patient data
- Enabling data-driven execution of MLMs
- Implementing the ArdenHostInterface
- Implementing a GUI and Web application for execution of user driven MLMs
- Implementing mechanisms to present information generated by MLMs

5.2 Connecting PDMS and Arden engine

MLMs can access patient data from the proxyDB that is periodically updated by the export handler. The replication interval is 10 min. The export handler also creates an event list that is processed by an integrated web service client. Our implementation of the ArdenHostInterface contains a control method for production mode to process clinical events and another for development of MLMs. We implemented email addresses, as well as DECT phone numbers for SMS messages, as communication endpoints.

Multi-ICU support is shown in Fig. 2. Each ICU has its own export VM, running an export job and an export handler. The PDMS exporter is capable of detecting changes in patient records. Any change results in a complete update of the specific patient record ("full snapshot").

The GUI of our MLM viewer is deliberately reduced to a minimum. It consists of a browser element and two buttons for printing and exiting. The browser element displays a web application consisting of a row of buttons and a display area. Clicking a button causes the integrated web service client to send an event message to the Arden Server. All resulting output is generated by MLMs, no refining is done by the web application or the MLM-viewer. Mapping of buttons to MLMs is of m:n type.

5.3 Generating extended output formats

All user requirements regarding the presentation of results were solely implemented as MLMs that are called as subroutines. Formats of timestamps are transformed to equal those of the PDMS. Further refining can be done in any MLM by calling the following MLMs:

- *fontformatter*: Arbitrary formatting of text
- *list_to_table*: Plain lists are converted to tables

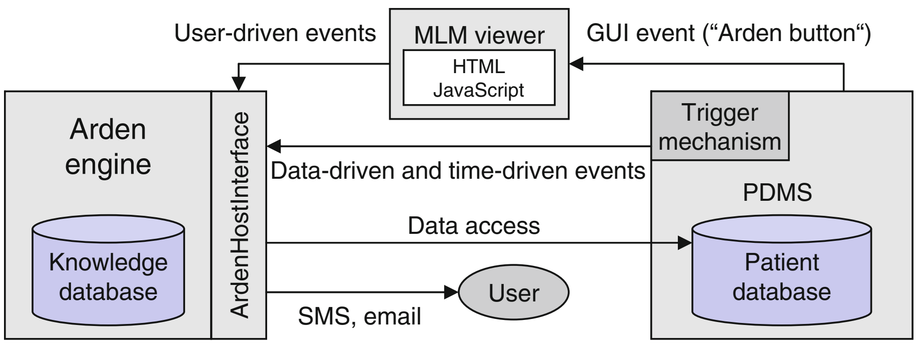


Fig. 1 Basic architecture of the intended system. Data-driven and time-driven events are directly sent to the Arden engine. User-driven events are triggered by buttons in the web application displayed by the MLM viewer. The viewer is started by a GUI event ("Arden button")

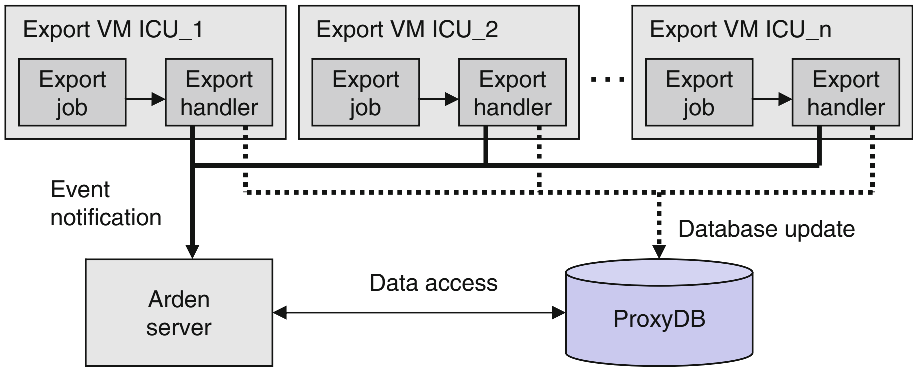


Fig. 2 Architecture of the multi-ICU capable system. Each ICU runs an export VM with one export job and one export handler. Database updates are performed as full snapshots of patient records. The export handlers detect events by comparing exports and notify the Arden server

- *tablegenerator*: Table description objects are converted to tables
- *hide_show*: Arbitrary items (e.g. tables) can be expanded and collapsed
- *lineplotter*: Scalable line plots can be generated from description objects

We strictly separated MLM output from the storage of derived information. Any return value is generally treated as GUI output. All additional derived information can be written to the proxyDB.

Figure 3 shows an example of the creation of a simple table from a list. This example has been taken from a production MLM generating lists of all critically altered calcium values of a patient stay. The table is created using a single line of code in the calling MLM. Calling *list_to_table* with a list of critical calcium values generates a table with three columns: a consecutive number, the value and its timestamp. This is done by looping through the list and chaining HTML tags to a table. The table is then returned by *list_to_table* as a single string.

Figure 4 shows an example for generating a line plot from a list of MELD score values (derived by an MLM) using a description object. The principle is the same as shown in Fig. 3, except it generates JavaScript instead of HTML. After instantiating the description object some properties are assigned. Calling the MLM *lineplotter* creates JavaScript code that draws a dynamic scalable line plot within the web browser. Points that correspond to single measurements are highlighted on mouse over. Some additional properties like borderlines or marking of single points could be set but are not shown in the example to keep it simple.

All presentation MLMs create single strings that can be concatenated at the end of the ACTION slot to the return value of the calling MLM and be displayed by any viewer that is capable of displaying HTML/JavaScript.

6 Discussion

We successfully integrated Arden based clinical decision support into our PDMS, enabling user-driven, time-driven and data-driven execution of MLMs. MLMs can call a number of presentation MLMs that may be extended in the future. The following aspects are to be discussed:

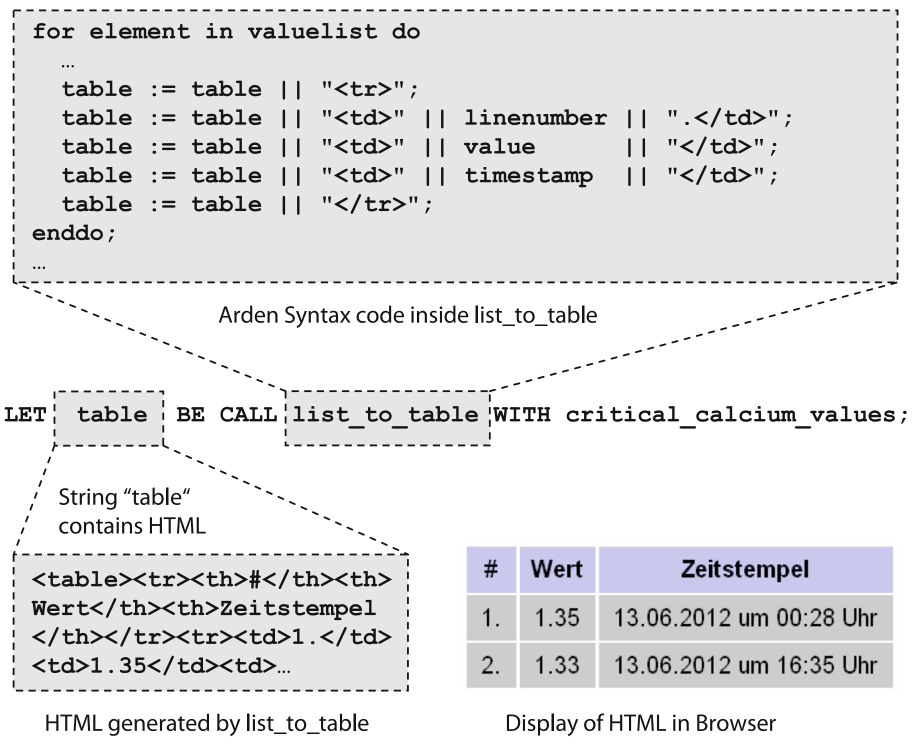
- Properties and drawbacks of the implemented platform
- Implementing presentation mechanisms in Arden Syntax
- Experiences with the Arden Syntax

6.1 Lessons learned from integrating Arden Syntax with a black box clinical information system

The technical integration was successful but some drawbacks remain. The first one affects the workaround to provide a data-driven trigger mechanism. Some authors describe the use of database triggers to provide data-driven execution of MLMs [17, 27]. We did not have this opportunity due to a lack of database access. A database trigger on the proxyDB instead of the patient database was also not possible as we update all patient data in the proxyDB by snapshots. Some cases are described where a database trigger cannot be implemented, involving the construction of an external component that detects events by scanning the database [28, 11]. We were completely limited to periodic exports due to lacking external database access and had to implement the event detection mechanism in the export handler. Our replication interval is a trade-off between system load and performance and is currently set to 10 min. Hence, the alert message for a critical event like a hypoglycaemia is delayed for up to 10 min after database entry. This delay could be reduced still further, however only as far as the resulting system load would allow (in our system about 5 min).

The replication delay also affects the content a user sees in the MLM viewer. If a user sees a new value in the GUI of the PDMS and starts an MLM that would process this value, it can take up to ten minutes until the MLM is able to read this value because it is only available after the next replication step.

Another drawback is the high consumption of system resources. One heavily utilized export VM is needed for



```
for element in valuelist do
  ...
  table := table || "<tr>";
  table := table || "<td>" || linenumber || ".</td>";
  table := table || "<td>" || value      || "</td>";
  table := table || "<td>" || timestamp  || "</td>";
  table := table || "</tr>";
enddo;
...
```

```
LET table BE CALL list_to_table WITH critical_calcium_values;
```

```
<table><tr><th>#</th><th>
Wert</th><th>Zeitstempel
</th></tr><tr><td>1.</td>
<td>1.35</td><td>...
```

#	Wert	Zeitstempel
1.	1.35	13.06.2012 um 00:28 Uhr
2.	1.33	13.06.2012 um 16:35 Uhr

Arden Syntax code inside list_to_table
String "table" contains HTML
HTML generated by list_to_table
Display of HTML in Browser

Fig. 3 Example for generating a table from a list. The presentation MLM *list_to_table* returns a string containing an HTML table with three columns: consecutive number, value and timestamp. The table is a screenshot from a real patient. More complex tables can be generated from description objects using the presentation MLM *tablegenerator*

```
LET dd          BE NEW diagramdescription;
LET dd.label    BE "Meldscore";
LET dd.values   BE meldscorelist;
LET dd.fromdate BE stay.admissiondate;
LET dd.todate   BE NOW;
LET diagram     BE CALL lineplotter WITH dd;
```

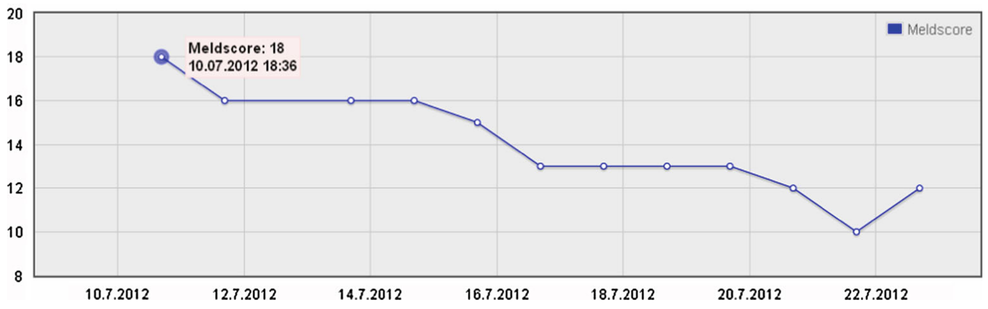


Fig. 4 MELD score: example for generating a line plot from a description object. The presentation MLM *lineplotter* returns a string containing JavaScript code that draws a scalable line plot with highlighting of points on mouse-over events. The graph is a screenshot from a real patient

any ICU to export patient data. These machines cannot be used for other purposes in export mode. Additional system load is created by constant updates of the proxyDB and constant checks for new events by the export handler. At least the system load is somewhat balanced because of the distributed architecture. The cost of wasted resources is high for substituting interfaces that ideally should be provided by the PDMS. The curly braces problem itself proved not to be difficult to solve and only required normal programming work.

6.2 Presentation mechanisms implemented in Arden Syntax

We encountered strong demand for the presentation of MLM results in formats other than plain text. This matter is rarely addressed in publications regarding the Arden Syntax. The specification of our used version 2.5 [29] only contains some string operators like the use of format strings. This is not surprising for two reasons: Firstly, a message created by an MLM usually contains only a short text. A typical example from Henri Mondor Hospital [11]:

"serum potassium (3.3) is low (minimum is 3.5)"

Secondly, MLM-generated strings are usually sent to communication endpoints that are not capable of displaying formats other than plain text (pagers, DECT phones, alert boxes). Regarding mobile devices in the clinical setting this is likely to change in the near future as for example modern mobile phones start to process HTML.

Karlsson et al. [30] describe a web based telemedical platform where strings returned by MLMs are inserted into a HTML template. Our MLMs in contrast return strings that already contain HTML or JavaScript generated by presentation-MLMs. Both approaches can be combined. Advanced presentation of information can easily be provided using templates in HTML and JavaScript. Such templates can be developed using a standard browser and than integrated in an MLM that adds patient data to the template.

Manual insertion of HTML in a generated string is considered unsuitable as it obliterates the code, complicates maintenance and stresses the knowledge engineer. Leaving the burden of coding HTML in MLMs to the knowledge engineer contradicts the paradigm of doctors as programmers and most likely would reduce the willingness to write more complex MLMs.

Of course, there are alternative approaches to present information. For example, MLMs could generate XML by calling a mapper on an Arden Syntax object. The XML data could be transformed to HTML/Javascript by a stylesheet. Which of these approaches is easier for a particular user depends on his individual skills. We presume that the presentation MLMs could be used more easily by a physician acting as a knowledge engineer.

Another approach would be to refine the information by a presentation layer inside the system itself. However, such an approach would interfere with the design goal of knowledge transfer. If the process of refining is integrated into the presentation layer of the CDSS, it cannot easily be transferred with the MLMs. If it is implemented as presentation-MLMs, these can be transferred with the other MLMs containing the medical knowledge. A knowledge transfer requires only a normal web browser on the receiver side to view the MLM output in the same way as the sender does.

6.3 Experiences with the Arden Syntax

From a technical point of view any high-level programming language could have been used to implement the described functionality. We decided to use Arden Syntax for two reasons. Firstly, it is an HL7 standard that supports knowledge transfer. Secondly, it is a language specifically designed for the medical domain that is very easy to learn, even promising to allow physicians to implement decision support functions on their own.

Compared to some other MLM libraries (for example [31]) ours is rather small. However we were still able to gain valuable experience with the Arden Syntax that is sometimes described to have following limitations:

> Arden syntax has two key limitations: first, it can only be used to encode event-driven, patient-specific rules. For use cases such as drug–drug interaction checking, or panic lab value alerting, this modality is sufficient. However, because Arden Syntax is patient-specific, it cannot be used for population-based decision support (such as a quality-of-care dashboard), and because it is event-driven, it cannot be used for point-of-care reference or information retrieval support. [32]

It may be that such limitations occur in the case of some embedded commercial Arden environments, but we did not encounter them as we had full control over the ArdenHostInterface and could implement all methods at our convenience. The Arden Syntax as a language specification is not bound by the above limitations. The structure of MLMs is optimized for data-driven execution, but they can also be started by a button or using any other convenient method. Arden Syntax is also not necessarily patient specific. It is true that in the case of alerting MLMs are executed in the context of one single patient. But that is only due to the fact that the event message is in that patient's context, so data-driven execution of list-based MLMs would simply not make sense.

An MLM can also process a whole patient list (we implemented several MLMs that do so) or work population-based decision support like monitoring infection outbreaks. Some authors have suggested an adoption beyond classical alerting [33–35]. An example for the use of MLM-packages [36] with high complexity is described at the University Hospital of Vienna [37, 38].

The Arden Syntax is a turing-complete programming language, so it can be used for an arbitrary application range. Only a few of the MLMs we developed during the cooperation would fit the typical scope of the Arden Syntax. Some of our MLMs do not derive any information but only present list-based patient data or a different arrangement for customized information retrieval.

A high level of benefit comes from the object data type that was proposed by Jenders et al. [39] and introduced in version 2.5. It overcomes the limitations of the flat lists that were the only complex data types before and did not support element access by name. As objects can contain lists and vice versa, the capabilities of the Arden Syntax have strongly increased and passing complex arguments between MLMs is made much easier as demonstrated in the line plotter example.

6.4 Generalizability of our technical approach

Our solution demonstrates that it is possible to connect an Arden Engine to a commercial clinical information system without direct database access or a trigger mechanism, presupposed this system provides an exporting mechanism. The drawbacks include delays in event detection and data provision in proportion to the configured replication interval, as well as a substantial system load caused by the continuous exports. Although our solution is tailored to our proprietary exporter, the basic approach of periodic replication into an external database appears also workable at other institutions. The integration of an Arden engine in any institution with any clinical information system requires the implementation of an interface component like the ArdenHostInterface, as the Arden Syntax standard generally leaves this step to the particular institution. Our presentation MLMs are fully generalizable, insofar as they do not introduce new dependencies on another specific technology. These MLMs could be transferred to any institution that uses Arden Syntax, without requiring adjustment as the only prerequisite is any modern web browser.

7 Conclusion

We have found that the Arden Syntax is suitable for far more than data-driven alerting. Since the introduction of objects in version 2.5, the capabilities of the Arden Syntax combined with its easy-to-learn syntax could even make it a suitable generic plug-in language for clinical information systems. It could be seen in a broader context as tool for providing clinical information systems with additional capabilities. Our main problem in integrating Arden Syntax into our PDMS was its lack of adequate interfaces. As long as it does not provide them we are limited to workarounds like replicating patient data into the proxyDB. Until now we have not been writing back MLM-generated data into the patient record, but have stored it in the proxyDB. Currently, we are working on enabling MLMs to write data into the patient record using the PDMS inbound interfaces. We also intend to create a chain of escalation to ensure prompt reactions to critical events.

References

1. Pryor TA, Hripcsak G. The Arden Syntax for medical logic modules. Int J Clin Monit Comput. 1993;10(4):215–24.
2. Kawamoto K, Houlihan CA, Balas EA, Lobach DF. Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success. BMJ. 2005;330(7494):765.
3. Randell R, Mitchell N, Dowding D, Cullum N, Thompson C. Effects of computerized decision support systems on nursing performance and patient outcomes: a systematic review. J Health Serv Res Policy. 2007;12(4):242–51.
4. Shojania KG, Jennings A, Mayhew A, Ramsay C, Eccles M, Grimshaw J. Effect of point-of-care computer reminders on physician behaviour: a systematic review. Can Med Assoc J. 2010;182(5):E216–25.
5. Jaspers MWM, Smeulers M, Vermeulen H, Peute LW. Effects of clinical decision-support systems on practitioner performance and patient outcomes: a synthesis of high-quality systematic review findings. J Am Med Inform Assoc. 2011;18(3):327.
6. Garg AX, Adhikari NKJ, McDonald H, Rosas-Arellano MP, Devereaux P, Beyene J, Sam J, Haynes RB. Effects of computerized clinical decision support systems on practitioner performance and patient outcomes. JAMA. 2005;293(10):1223–38.
7. 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(2):135–42.
8. Hunt DL, Haynes RB, Hanna SE, Smith K. Effects of computer-based clinical decision support systems on physician performance and patient outcomes: a systematic review. JAMA. 1998;280(15):1339–46.
9. Hripcsak G, Ludemann P, Pryor TA, Wigertz OB, Clayton PD. Rationale for the Arden Syntax. Comput Biomed Res. 1994;27(4):291–324.
10. Hripcsak G. Writing Arden Syntax medical logic modules. Comput Biol Med. 1994;24(5):331–63.
11. Karadimas HC, Chailloleau C, Hemery F, Simonnet J, Lepage E. Arden/J: an architecture for MLM execution on the java platform. J Am Med Inform Assoc. 2002;9(4):359–68.
12. Gietzelt M, Goltz U, Grunwald D, Lochau M, Marschollek M, Song B, Wolf KH. ARDEN2BYTECODE: a one-pass Arden Syntax compiler for service-oriented decision support systems based on the OSGi platform. Comput Methods Programs Biomed. 2012;106(2):114–25. doi:10.1016/j.cmpb.2011.11.003.
13. Fehre K, Mandl H, Adlassnig KP. Service-Oriented, Arden-Syntax-Based clinical decision support. In: Proceedings of eHealth2011. Austrian Computer Society; 2011. p. 123–8.
14. Sojer R, Bürkle T, Criegee-Rieck M, Neubert A, Brune K, Prokosch HU. Knowledge modelling and knowledge representation in hospital information systems to improve drug safety. J Inf Technol Healthc. 2006;29.
15. Gao X, Johansson B, Shahsavar N, Arkad K, Åhlfeldt H, Wigertz O. Pre-compiling medical logic modules into C++ in building medical decision support systems. Comput Methods Programs Biomed. 1993;41(2):107–19.
16. Kuhn RA, Reider RS. A C++ framework for developing medical logic modules and an Arden Syntax compiler. Comput Biol Med. 1994;24(5):365–70.
17. Tafazzoli AG, Altmann U, Wachter W, Katz FR, Holzer S, Dudeck J. Integrated knowledge-based functions in a hospital cancer registry–specific requirements for routine applicability. Proc Amia Symp; 1999. p. 410–4. http://www.ncbi.nlm.nih.gov/pubmed/10566391.
18. Liang YC, Chang P. The development of variable MLM editor and TSQL translator based on Arden Syntax in Taiwan. In: Proceedings of AMIA annual symposium; 2003. p. 908.
19. Kim S, Haug PJ, Rocha RA, Choi I. Modeling the Arden Syntax for medical decisions in XML. Int J Med Inf. 2008;77(10):650–6.
20. Jung CY, Sward KA, Haug PJ. Executing medical logic modules expressed in ArdenML using drools. J Am Med Inform Assoc. 2012;19(4):533–6. doi:10.1136/amiajnl-2011-000512.

21. Oppenheim MI, Mintz RJ, Boyer AG, Frayer WW. Design of a clinical alert system to facilitate development, testing, maintenance, and user-specific notification. In: Proceedings of AMIA symposium; 2000. p. 630–4.
22. Hripcsak G, Clayton PD, Jenders RA, Cimino JJ, Johnson SB. Design of a clinical event monitor. Comput Biomed Res. 1996;29(3):194–221.
23. Gietzelt M, Goltz U, Grunwald D, Lochau M, Marschollek M, Song B, Wolf KH. Arden2ByteCode: a one-pass Arden Syntax compiler for service-oriented decision support systems based on the OSGi platform. Comput Methods Programs Biomed; 2011.
24. Greenes RA. Clinical decision support: the road ahead. Amsterdam: Academic Press; 2007.
25. Burkle T, Castellanos I, Tech H, Prokosch HU. Implementation of a patient data management system—an evaluation study of workflow alterations. Stud Health Technol Inform. 2010;160 (Pt 2):1256–60.
26. Nadkarni PM. Metadata-driven Software Systems in Biomedicine Designing Systems that can adapt to changing knowledge: health informatics. London: Springer-Verlag London Limited; 2011.
27. Arkad K, Ahlfeldt H, Gao X, Shahsavar N, Wigertz O, Jean FC, Degoulet P. Integration of data driven decision support into the HELIOS environment. Int J Biomed Comput. 1994;34(1):195–205.
28. Johansson B, Bergqvist Y. Integrating decision support, based on the Arden Syntax, in a clinical laboratory environment. In: Proceedings of annual symposium on computers applied to medical care; 1993. p. 394–8. http://www.ncbi.nlm.nih.gov/pubmed/8130502.
29. V2.5-2005 AHA. Health Level Seven Arden Syntax, Version 2.5 (revision of ANSI/HL7 Arden V2.1-2002). 2005.
30. Karlsson D, Ekdahl C, Wigertz O, Shahsaver N, Gill H, Forsum U. Extended telemedical consultation using Arden Syntax based decision support, hypertext and WWW technique. Methods Inf Med. 1997;36(2):108–14.
31. Jenders R, Huang H, Hripcsak G, Clayton P. Evolution of a knowledge base for a clinical decision support system encoded in the Arden Syntax. In: Proceedings of AMIA Symposium American Medical Informatics Association; 1998. p. 558–62.
32. Wright A, Sittig DF. A four-phase model of the evolution of clinical decision support architectures. Int J Med Inf. 2008;77(10): 641–9.
33. Sailors RM, Bradshaw RL, East TD Moving Arden Syntax outside of the (Alert) box: a paradigm for supporting multi-step clinical protocols. In: Proceedings of AMIA Symposium; 1998. p. 1071.
34. Sherman EH, Hripcsak G, Starren J, Jenders RA, Clayton P. Using intermediate states to improve the ability of the Arden Syntax to implement care plans and reuse knowledge. In: Proceedings of AMIA annual; 1995. p. 238–42.
35. Starren J, Hripcsak G, Jordan D, Allen B, Weissman C, Clayton P. Encoding a post-operative coronary artery bypass surgery care plan in the Arden Syntax. Comput Biol Med. 1994;24(5):411–7.
36. Adlassnig KP, Rappelsberger A. Medical knowledge packages and their integration into health-care information systems and the World Wide Web. Stud Health Technol Inform. 2008;136:121–6.
37. Koller W, Blacky A, Bauer C, Mandl H, Adlassnig KP. Electronic surveillance of healthcare-associated infections with MONI-ICU–a clinical breakthrough compared to conventional surveillance systems. Stud Health Technol Inform. 2010;160 (Pt 1):432–6.
38. Adlassnig KP, Blacky A, Koller W. Artificial-intelligence-based hospital-acquired infection control. Stud Health Technol Inform. 2009;149:103–10.
39. Jenders RA, Corman R, Dasgupta B. Making the standard more standard: a data and query model for knowledge representation in the Arden syntax. In: Proceedings of AMIA annual symposium; 2003. p. 323–30.